ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΥΠΟΥΡΓΕΙΟ ΕΣΩΤΕΡΙΚΩΝ ΑΠΟΚΕΝΤΡΩΣΗΣ & ΗΛΕΚΤΡΟΝΙΚΗΣ ΔΙΑΚΥΒΕΡΝΗΣΗΣ

ΥΠΟΥΡΓΕΙΟ ΕΣΩΤΕΡΙΚΩΝ
ΕΘΝΙΚΟ ΚΕΝΤΡΟ ΔΗΜΟΣΙΑΣ ΔΙΟΙΚΗΣΗΣ & ΑΥΤΟΔΙΟΙΚΗΣΗΣ

ΕΣΠΑ 2007-2013 πρόγραμμα για την ανάπτυξη

Ε.Π. ΔΙΟΙΚΗΤΙΚΗ ΜΕΤΑΡΡΥΘΜΙΣΗ

ΙΝΣΤΙΤΟΥΤΟ ΕΠΙΜΟΡΦΩΣΗΣ

ΤΙΤΛΟΣ ΕΠΙΜΟΡΦΩΤΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ:

Προγραμματισμός Λογιστικών Φύλλων

Συγγραφή: Θ. Τρίμμης- Κόρος

Επ. Υπεύθυνος: Δρ. Ηλίας Μαραγκός

Αθήνα 2010

Περιεχόμενα

Μακροεντολές

Τι είναι οι μακροεντολές

Κατά τη χρήση του Excel εμφανίζονται σειρές εντολών ή ενεργειών τις οποίες εκτελούμε συχνά και επαναλαμβανόμενα, όπως για παράδειγμα η εφαρμογή μιας συγκεκριμένης μορφοποίησης σε κελιά ή/και κείμενο. Προκύπτει έτσι συχνά η ανάγκη αυτοματοποίησης και ομαδοποίησης των ενεργειών αυτών ώστε να μπορούμε να έχουμε το αποτέλεσμα που θέλουμε εύκολα και γρήγορα χωρίς να χρειάζεται κάθε φορά να τις επαναλαμβάνουμε όλες. Ακόμα, η αυτοματοποίηση της διαδικασίας είναι επιθυμητή και σε περιπτώσεις όπου πραγματοποιούμε έναν πολύπλοκο υπολογισμό ή εκτελούμε μια μεγάλη ακολουθία εντολών, έστω και αν δεν είναι από τις ενέργειες που κάνουμε συχνά.

Τα παραπάνω μπορούν να επιτευχθούν με δύο συμπληρωματικούς τρόπους: α) Με τη καταγραφή και χρήση **Μακροεντολών (Macros)** και β) Με τον προγραμματισμό μέσω της **VBA (Visual Basic for Applications)**.

Οι μακροεντολές καταγράφουν την ακολουθία εντολών και ενεργειών του χρήστη μέσα από λειτουργία που ενεργοποιείται από το σχετικό μενού του Excel (Εργαλεία/Μακροεντολή). Στη συνέχεια τις ανακαλούμε και τις εκτελούμε όποτε μας χρειάζονται, με το όνομα με το οποίο τις έχουμε αποθηκεύσει.

Μία μακροεντολή απαρτίζεται από εντολές σε κώδικα VBA οι οποίες δημιουργούνται την ώρα της καταγραφής της. Μέσω του **Επεξεργαστή Visual Basic (VB Editor)** είναι δυνατό να γίνουν τροποποιήσεις και συμπληρώσεις στις καταγεγραμμένες εντολές, δίνοντάς τους περισσότερη λειτουργικότητα και μεγαλύτερες δυνατότητες.

Τι μπορούμε να κάνουμε με τις μακροεντολές

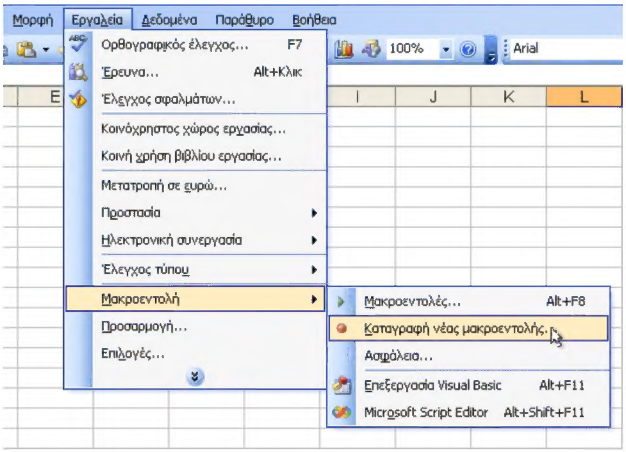
Συνοψίζοντας τα παραπάνω, με τις μακροεντολές μπορούμε:

- Να αυτοματοποιήσουμε μια σειρά πολύπλοκων εντολών – ενεργειών. Για παράδειγμα, μπορούμε να δημιουργούμε ένα πίνακα με τις ίδιες πάντα επικεφαλίδες και μορφοποίηση, ή να εισάγουμε εύκολα ένα πολύπλοκο τύπο σε πολλά κελιά
- Να αυτοματοποιήσουμε μια σειρά απλών εντολών – ενεργειών που όμως επαναλαμβάνουμε συχνά. Για παράδειγμα, αν θέλουμε να επαναλαμβάνουμε μία συγκεκριμένη μορφοποίηση σε πολλά κελιά, σε διαφορετικά φύλλα ή βιβλία εργασίας
- Να δημιουργήσουμε μία δική μας εντολή (που όμως θα αποτελείται από συνδυασμό εντολών του Excel), την οποία θα εκτελούμε με κάποιο πλήκτρο συντόμευσης ή μέσω ενός εικονιδίου ή στοιχείου μενού που θα έχουμε προσθέσει στις γραμμές εργαλείων ή στη γραμμή μενού.

Καταγραφή μακροεντολών

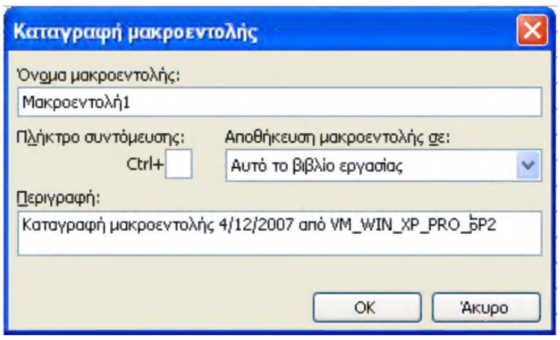
Προκειμένου να καταγράψουμε μία μακροεντολή ακολουθούμε τα παρακάτω βήματα:

1. Από τη γραμμή μενού επιλέγουμε τη διαδρομή **Εργαλεία – Μακροεντολή – Καταγραφή νέας μακροεντολής**



Εικόνα 1 Έναρξη καταγραφής μίας μακροεντολής από το μενού

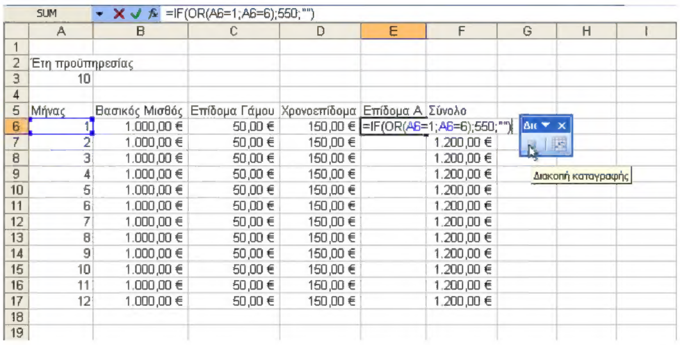
2. Στο παράθυρο διαλόγου που εμφανίζεται:
    a. πληκτρολογούμε το όνομα με το οποίο θέλουμε να αποθηκεύσουμε την μακροεντολή
    b. επιλέγουμε αν θα αποθηκεύσουμε την μακροεντολή στο ενεργό βιβλίο εργασίας, σε νέο βιβλίο ή στο βιβλίο προσωπικών μακροεντολών
    c. δηλώνουμε αν επιθυμούμε κάποιο πλήκτρο συντόμευσης, ώστε να μπορούμε να εκτελούμε την μακροεντολή γρήγορα
    d. περιγράφουμε, αν το επιθυμούμε, τη μακροεντολή



Εικόνα 2 Το παράθυρο Καταγραφή μακροεντολής

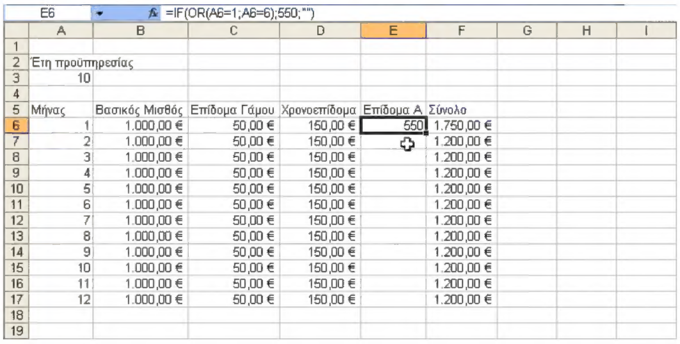
Μόλις πατήσουμε το πλήκτρο **OK** κλείνει αυτό το παράθυρο και εμφανίζεται η εργαλειοθήκη **Διακοπή καταγραφής** (εικόνα 3)

3. Εκτελούμε τις ενέργειες που θέλουμε να καταγράψουμε στη μακροεντολή



Εικόνα 3 Εκτελούμε τις εντολές που θέλουμε να καταγράψουμε στη μακροεντολή

4. Πατάμε το πλήκτρο **Διακοπή καταγραφής** στην αντίστοιχη εργαλειοθήκη, για να σταματήσει η καταγραφή και να αποθηκευτεί η μακροεντολή. Η εργαλειοθήκη Διακοπή καταγραφής εξαφανίζεται και επιστρέφουμε στην συνήθη κατάσταση λειτουργίας.

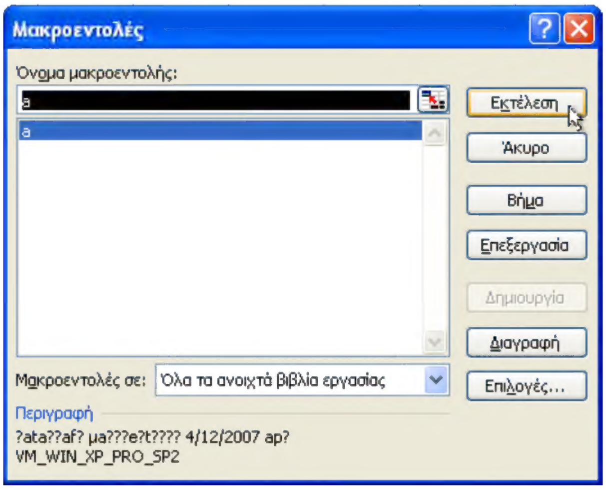


Εικόνα 4 Η μακροεντολή έχει καταγραφεί

Η μακροεντολή έχει πλέον αποθηκευτεί και <u>είναι διαθέσιμη για να την εκτελέσουμε</u> όποτε το επιθυμούμε.

Μπορούμε να δούμε τις μακροεντολές που καταγράψαμε και γενικότερα όλες τις αποθηκευμένες μακροεντολές από την επιλογή της γραμμής μενού **Εργαλεία – Μακροεντολή – Μακροεντολές** ή με τη χρήση των πλήκτρων συντόμευσης **Alt+F8**. Στο παράθυρο που ανοίγει έχουμε τη δυνατότητα:

- να επιλέξουμε μία μακροεντολή,
- να την εκτελέσουμε ολόκληρη,
- να την εκτελέσουμε βηματικά,
- να την διαγράψουμε,
- να την επεξεργαστούμε,
- να της αναθέσουμε πλήκτρο συντόμευσης ή να τροποποιήσουμε την περιγραφή της (πατώντας το πλήκτρο **Επιλογές**), καθώς και
- να δημιουργήσουμε μία νέα μακροεντολή.



Εικόνα 5 Το παράθυρο Μακροεντολές από το μενού Εργαλεία - Μακροεντολή

Εκτέλεση μακροεντολών

Το Excel μας παρέχει διάφορους τρόπους εκτέλεσης μίας μακροεντολής.

Εκτέλεση μακροεντολής από το γραμμή μενού

Από την επιλογή της γραμμής μενού **Εργαλεία – Μακροεντολή – Μακροεντολές**, ανοίγουμε το παράθυρο **Μακροεντολές**, επιλέγουμε τη μακροεντολή που επιθυμούμε και πατάμε το πλήκτρο **Εκτέλεση**.

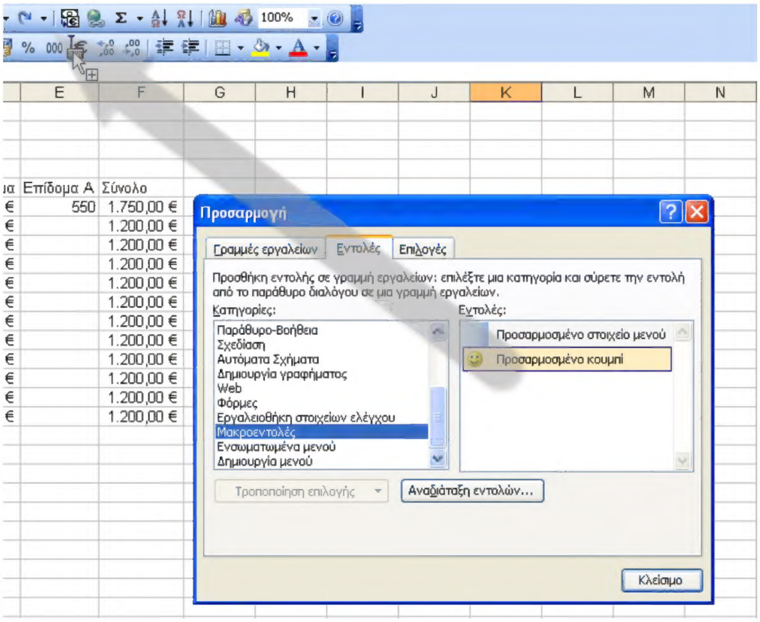
Εκτέλεση μακροεντολής με συνδυασμό πλήκτρων

Χρησιμοποιώντας τον σχετικό συνδυασμό πλήκτρων συντόμευσης που πρέπει προηγουμένως να έχουμε δηλώσει/ορίσει στο παράθυρο **Καταγραφή μακροεντολής**, ή στο παράθυρο **Επιλογές μακροεντολής**.

Εκτέλεση μακροεντολής από εικονίδιο στη γραμμή εργαλείων

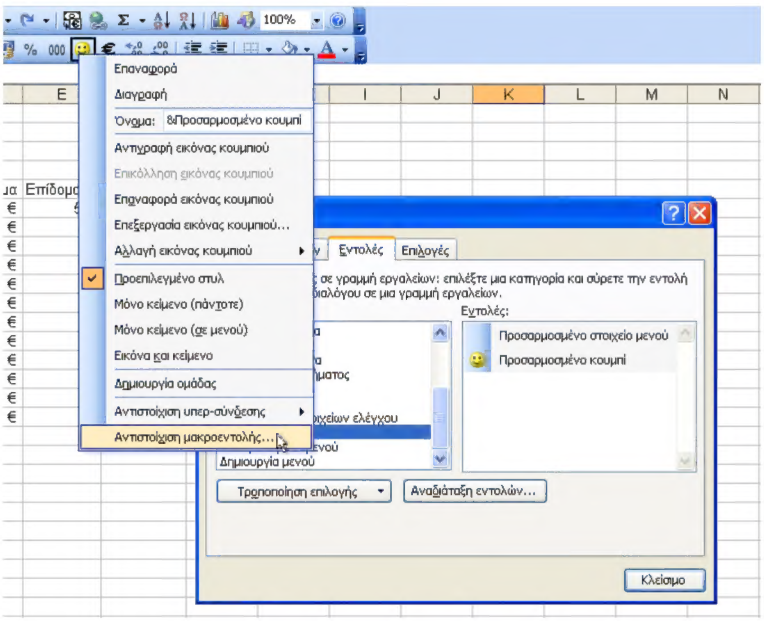
Μπορούμε να αντιστοιχίσουμε μία μακροεντολή σε εικονίδιο που θα προσθέσουμε στη γραμμή εργαλείων, για να την εκτελούμε εύκολα, ως εξής:

1. Από τη γραμμή μενού επιλέγουμε τη διαδρομή **Εργαλεία – Προσαρμογή.**
2. Στο παράθυρο **Προσαρμογή** επιλέγουμε την ετικέτα **Εντολές.**
3. Στις **Κατηγορίες** επιλέγουμε **Μακροεντολές.**



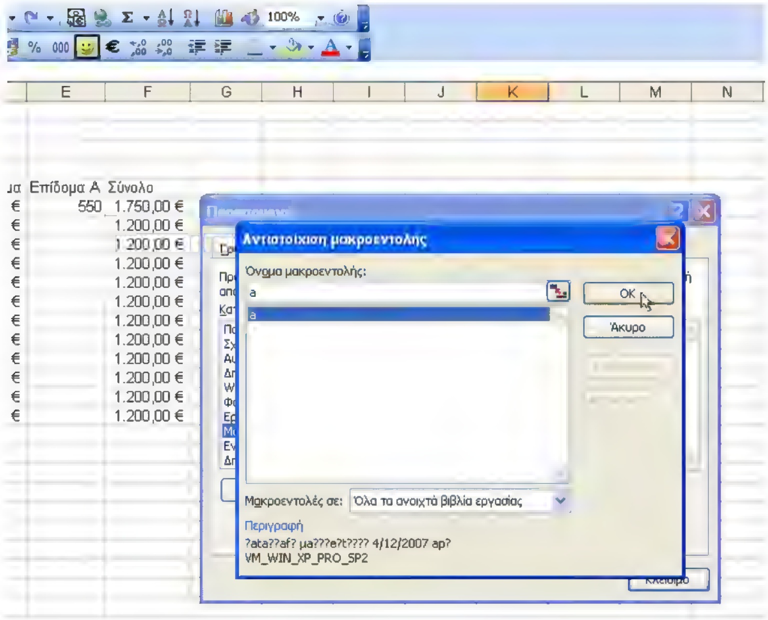
Εικόνα 6 Προσθήκη εικονιδίου στη γραμμή εργαλείων

4. Από τη λίστα ¨Εντολές¨ πατάμε με το αριστερό πλήκτρο του ποντικιού στην επιλογή **Προσαρμοσμένο κουμπί** και τη σύρουμε στο σημείο της γραμμής εργαλείων που θέλουμε να προσθέσουμε το εικονίδιο το οποίο θα καλεί τη μακροεντολή.
5. Με το ποντίκι πάνω στο εικονίδιο πατάμε το δεξί πλήκτρο και από το μενού που εμφανίζεται επιλέγουμε την εντολή **Αντιστοίχιση μακροεντολής.**



Εικόνα 7 Μενού επιλογών νέου εικονιδίου

6. Στο παράθυρο **Αντιστοίχιση μακροεντολής** επιλέγουμε τη μακροεντολή που θέλουμε και πατάμε **OK**.

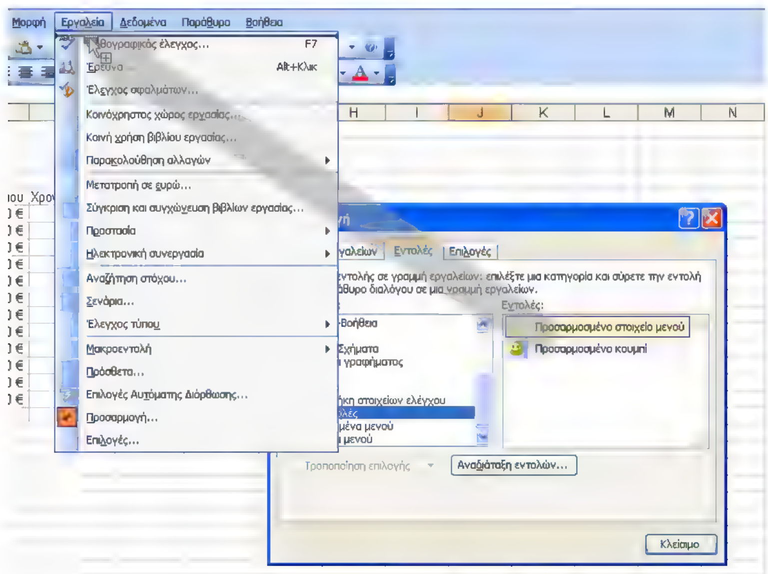


Εικόνα 8 Αντιστοίχιση μακροεντολής στο νέο εικονίδιο

Εκτέλεση μακροεντολής από εντολή μενού

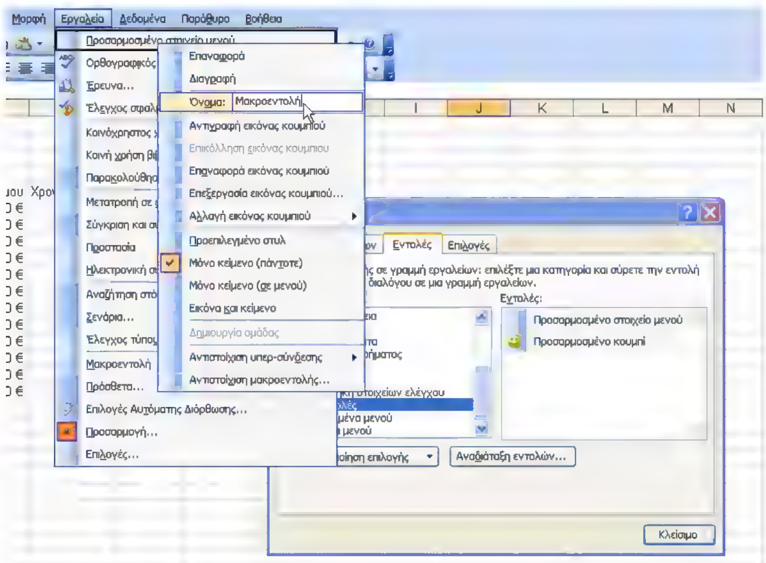
Μπορούμε να αντιστοιχίσουμε μία μακροεντολή σε εντολή ενός μενού ως εξής:

1. Από τη γραμμή μενού επιλέγουμε τη διαδρομή **Εργαλεία – Προσαρμογή.**
2. Στο παράθυρο **Προσαρμογή** επιλέγουμε την ετικέτα **Εντολές.**
3. Στις **Κατηγορίες** επιλέγουμε **Μακροεντολές.**
4. Πατάμε με το αριστερό πλήκτρο του ποντικιού στην εντολή **Προσαρμοσμένο** στοιχείο μενού και τη σύρουμε στο σημείο του μενού που θέλουμε να προσθέσουμε το στοιχείο που θα καλεί τη μακροεντολή.



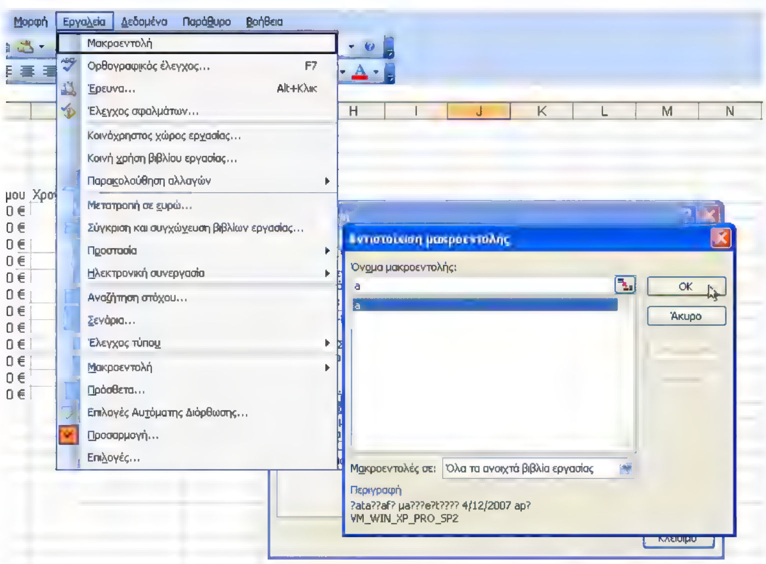
Εικόνα 9 Προσθήκη στοιχείου σε μενού

5. Με το ποντίκι πάνω στο στοιχείο πατάμε το δεξί πλήκτρο και από το μενού που εμφανίζεται επιλέγουμε την εντολή **Όνομα** για να τροποποιήσουμε το όνομα της εντολής και μετά επιλέγουμε την εντολή **Αντιστοίχιση μακροεντολής.**



Εικόνα 10 Μενού επιλογών νέου στοιχείου και τροποποίηση ονόματος

6. Στο παράθυρο **Αντιστοίχιση μακροεντολής** επιλέγουμε τη μακροεντολή που θέλουμε και πατάμε **ΟΚ**.
7. Στο παράθυρο **Προσαρμογή** πατάμε **Κλείσιμο**.



Εικόνα 11 Αντιστοίχιση μακροεντολής στο νέο στοιχείο

Διαγραφή μακροεντολής από εντολή μενού

Για να διαγράψουμε τη μακροεντολή επιλέγουμε **Εργαλεία, Προσαρμογή**. Κάνουμε δεξί κλικ στο **στοιχείο μενού (μακροεντολή)** που θέλουμε να διαγράψουμε. Από τον πίνακα εντολών που εμφανίζεται επιλέγουμε **Διαγραφή.** Πατώντας **Κλείσιμο ή Esc** επιστρέφουμε στο φύλλο.

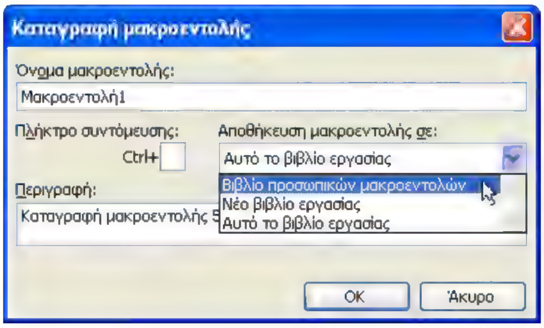
Το βιβλίο προσωπικών μακροεντολών (personal.xls)

Συνήθως καταγράφουμε τις μακροεντολές στο ενεργό βιβλίο εργασίας. Στην περίπτωση αυτή οι μακροεντολές μπορούν να εκτελεστούν μόνο μέσα στο συγκεκριμένο βιβλίο. Αν το κλείσουμε, δεν έχουμε πλέον δυνατότητα πρόσβασης στις μακροεντολές που περιέχει. Τι πρέπει λοιπόν να κάνουμε στην περίπτωση όπου επιθυμούμε να εκτελούμε συγκεκριμένες μακροεντολές από οποιοδήποτε βιβλίο και να έχουμε ανοικτό;

Το Excel, όπως είδαμε, μας παρέχει τρεις επιλογές για την αποθήκευση των μακροεντολών μας:

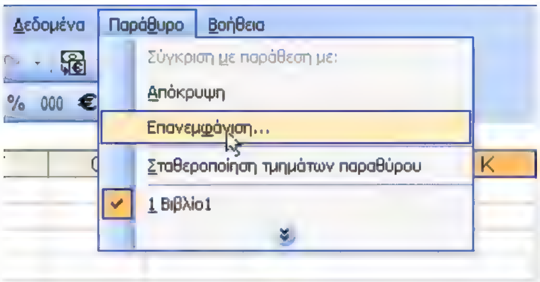
1. το ενεργό βιβλίο εργασίας,
2. ένα νέο βιβλίο εργασίας και
3. το **βιβλίο προσωπικών μακροεντολών (personal.xls)**.

Αν επιλέξουμε να αποθηκεύσουμε μία μακροεντολή στο τελευταίο, τότε αυτή μπορεί να εκτελεστεί μέσα από οποιοδήποτε βιβλίο εργασίας.



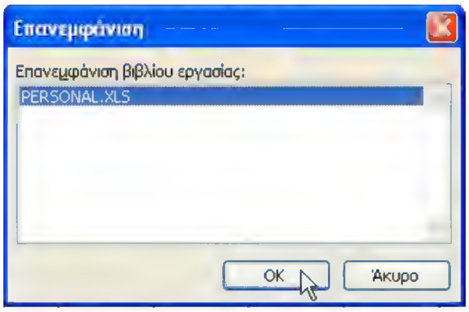
Εικόνα 12 Αποθήκευση μακροεντολής στο βιβλίο προσωπικών μακροεντολών

Το Excel δημιουργεί το βιβλίο προσωπικών μακροεντολών την πρώτη φορά που επιλέγουμε να αποθηκεύσουμε σε αυτό μία μακροεντολή. Στη συνέχεια το ανοίγει ως κρυφό (hidden) βιβλίο κάθε φορά που τρέχει η εφαρμογή. Ο χρήστης δεν γνωρίζει ότι ένα κρυφό βιβλίο εργασίας είναι ανοικτό παρά μόνο αν κοιτάξει στο παράθυρο **Μακροεντολές**. Για να εμφανίσουμε το βιβλίο προσωπικών μακροεντολών πρέπει να επιλέξουμε την εντολή **Επανεμφάνιση** από το μενού **Παράθυρο**. Η επανεμφάνιση ενός κρυφού βιβλίου εργασίας είναι απαραίτητη προϋπόθεση για να διαγράψουμε μία εντολή που είναι αποθηκευμένη σε αυτό.



Εικόνα 13 Μενού Παράθυρο

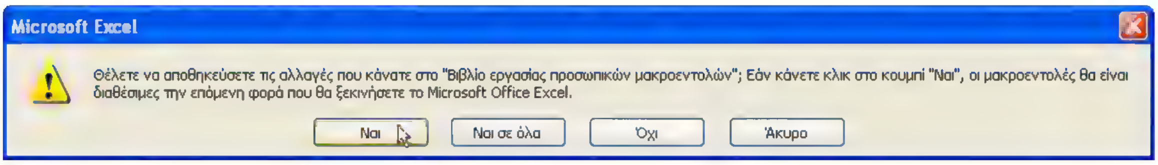
Στο παράθυρο που εμφανίζεται επιλέγουμε το κρυφό βιβλίο που θέλουμε να επανεμφανιστεί και πατάμε **OK**.



Εικόνα 14 Εμφάνιση κρυφού βιβλίου εργασίας

Από την εντολή **Απόκρυψη** μπορούμε να κάνουμε το ενεργό βιβλίο και πάλι κρυφό.

Όταν κλείνουμε την εφαρμογή, Το Excel μας ρωτάει αν επιθυμούμε να αποθηκεύσουμε τις αλλαγές που έχουμε κάνει στο βιβλίο προσωπικών μακροεντολών, ώστε να είναι διαθέσιμες την επόμενη φορά που θα εκκινήσουμε την εφαρμογή.



Εικόνα 15 Επιλογή αποθήκευσης βιβλίου προσωπικών μακροεντολών κατά την έξοδο από το Excel

Αυτόματη εκτέλεση μίας μακροεντολής κατά το άνοιγμα ενός βιβλίου εργασίας

Μία μακροεντολή μπορεί να εκτελεστεί αυτόματα κατά το άνοιγμα ενός βιβλίου εργασίας, εφόσον έχει αποθηκευτεί με το όνομα **Auto_Open**.

Όπως θα δούμε στη συνέχεια με τη VBA είναι δυνατόν να δημιουργήσουμε μακροεντολές που θα εκτελούνται όταν κάνουμε συγκεκριμένες ενέργειες, όπως για παράδειγμα όταν ενεργοποιούμε ή ανοίγουμε ένα συγκεκριμένο φύλλο εργασίας ή όταν δημιουργούμε ένα νέο φύλλο εργασίας.

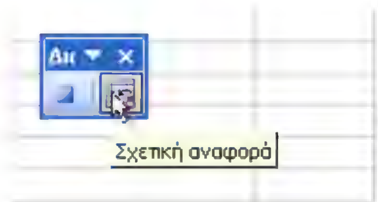
Πως εμποδίζουμε την αυτόματη εκτέλεση μακροεντολών κατά την εκκίνηση του Excel ή το άνοιγμα ενός βιβλίου εργασίας

Αν δεν επιθυμούμε την αυτόματη εκτέλεση της μακροεντολής Auto_Open που πιθανόν να περιέχει το προσωπικό βιβλίο εργασίας, τότε εκκινούμε το Excel πατώντας ταυτόχρονα το πλήκτρο Shift. Αν εκκινούμε το Excel από τη γραμμή συντομεύσεων του Microsoft Office τότε πρέπει να πατήσουμε το πλήκτρο του Excel και αμέσως μετά το πλήκτρο Shift.

Απόλυτες και σχετικές αναφορές κατά την καταγραφή μίας μακροεντολής

Κατά την καταγραφή μίας μακροεντολής μπορούμε να επιλέξουμε να χρησιμοποιούμε απόλυτες ή σχετικές αναφορές. Η επιλογή γίνεται από το κουμπί Σχετική αναφορά της εργαλειοθήκης Διακοπή καταγραφής.

- Αν θέλουμε οι καταγεγραμμένες εντολές να αναφέρονται και να ενεργούν στα ίδια κελιά (από πλευράς διευθύνσεων) κάθε φορά που εκτελούμε την μακροεντολή, τότε επιλέγουμε να καταγράφουμε χρησιμοποιώντας απόλυτες αναφορές.
- Αν όμως θέλουμε κάθε φορά οι εντολές να εκτελούνται σε σχέση με το ενεργό κελί τότε επιλέγουμε να καταγράφουμε χρησιμοποιώντας σχετικές αναφορές.



Εικόνα 16 Εργαλειοθήκη Διακοπή καταγραφής

Θεωρείστε, για παράδειγμα, ότι θέλουμε με μία μακροεντολή να εισάγουμε στα κελιά A1 ως A12 τους δώδεκα μήνες (Ιανουάριος, Φεβρουάριος ...). Στην περίπτωση αυτή θα πρέπει να χρησιμοποιήσουμε απόλυτες αναφορές. Αν όμως θέλουμε να εισάγουμε τους δώδεκα μήνες στο ενεργό κελί και στα ένδεκα επόμενα, κάτω από αυτό, τότε χρησιμοποιούμε σχετικές αναφορές κατά την καταγραφή της μακροεντολής.



Καταγραφή μακροεντολής με απόλυτες αναφορές



Καταγραφή μακροεντολής με σχετικές αναφορές

Εικόνα 17 Καταγραφή με χρήση απόλυτων ή σχετικών αναφορών

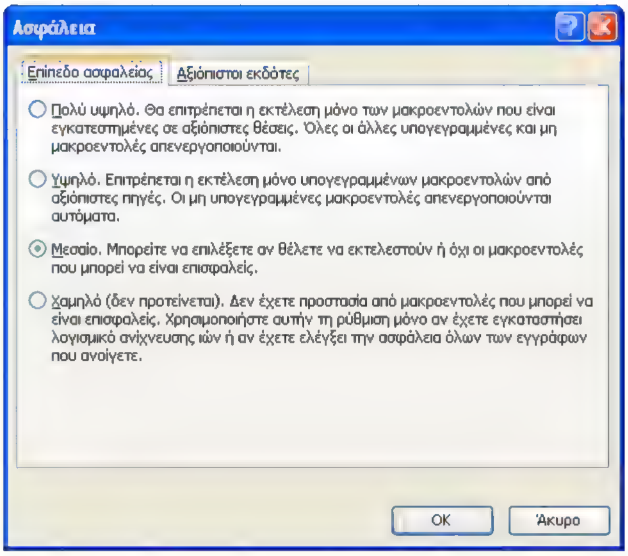
Ρύθμιση επιπέδου ασφαλείας μακροεντολών

Οι μακροεντολές είναι δυνατόν να περιέχουν εντολές επικίνδυνες σε σχέση με την ασφάλεια του υπολογιστή στον οποίο εκτελούνται. Για την αντιμετώπιση τέτοιων περιπτώσεων υπάρχει η δυνατότητα χρήσης ψηφιακών υπογραφών. Ο δημιουργός δηλαδή μίας μακροεντολής βεβαιώνει μέσω ψηφιακών πιστοποιητικών τη γνησιότητα των μακροεντολών που δημιουργεί. Η υπογραφή αυτή προσκολλάται στη μακροεντολή μέχρι η τελευταία να τροποποιηθεί, οπότε και απαιτείται εκ νέου η προσκόλληση υπογραφής, προκειμένου να βεβαιωθεί η γνησιότητα της νέας μακροεντολής.

Το Excel παρέχει τη δυνατότητα ρύθμισης της ασφαλείας, όσον αφορά στην εκτέλεση μακροεντολών που μπορεί να περιέχει ένα βιβλίο εργασίας, σε τέσσερα επίπεδα:

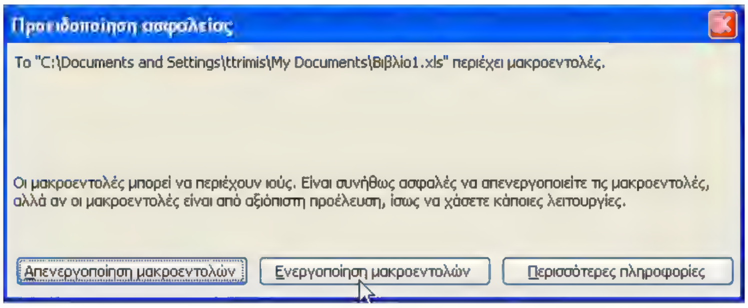
- Πολύ υψηλό: Επιτρέπεται η εκτέλεση μακροεντολών από αξιόπιστες θέσεις και μόνο.
- Υψηλό: Η ρύθμιση αυτή απενεργοποιεί τη δυνατότητα εκτέλεσης μη υπογεγραμμένων μακροεντολών, ακόμη και αυτών που εμείς δημιουργούμε. Παρέχει όμως τη δυνατότητα εκτέλεσης μακροεντολών από αξιόπιστες πηγές.
- Μεσαίο: Σε αυτό το επίπεδο ο χρήστης κατά το άνοιγμα ενός βιβλίου εργασίας έχει τη δυνατότητα επιλογής εκτέλεσης ή όχι των μακροεντολών (υπογεγραμμένων και μη).
- Χαμηλό: Το Excel ενεργοποιεί αυτόματα όλες τις μακροεντολές (υπογεγραμμένες και μη), χωρίς κανένα έλεγχο ή ερώτηση προς το χρήστη.

Η σχετική ρύθμιση γίνεται από την διαδρομή **Εργαλεία – Μακροεντολή – Ασφάλεια** της γραμμής μενού.



Εικόνα 18 Παράθυρο ρύθμισης επιπέδου ασφάλειας

Η ρύθμιση του επιπέδου ασφαλείας ισχύει για κάθε βιβλίο εργασίας που ανοίγουμε, χωρίς να είναι δυνατή η διαφορετική ρύθμιση ασφαλείας για κάθε βιβλίο. Αν η ρύθμιση ασφαλείας έχει καθοριστεί στο Μεσαίο επίπεδο τότε όταν ανοίγουμε ένα βιβλίο που περιέχει μακροεντολές εμφανίζεται η παρακάτω προειδοποίηση.



Εικόνα 19 Το μήνυμα που εμφανίζει το Excel, όταν ανοίγουμε ένα βιβλίο που περιέχει μακροεντολές (επίπεδο ασφάλειας: μεσαίο)

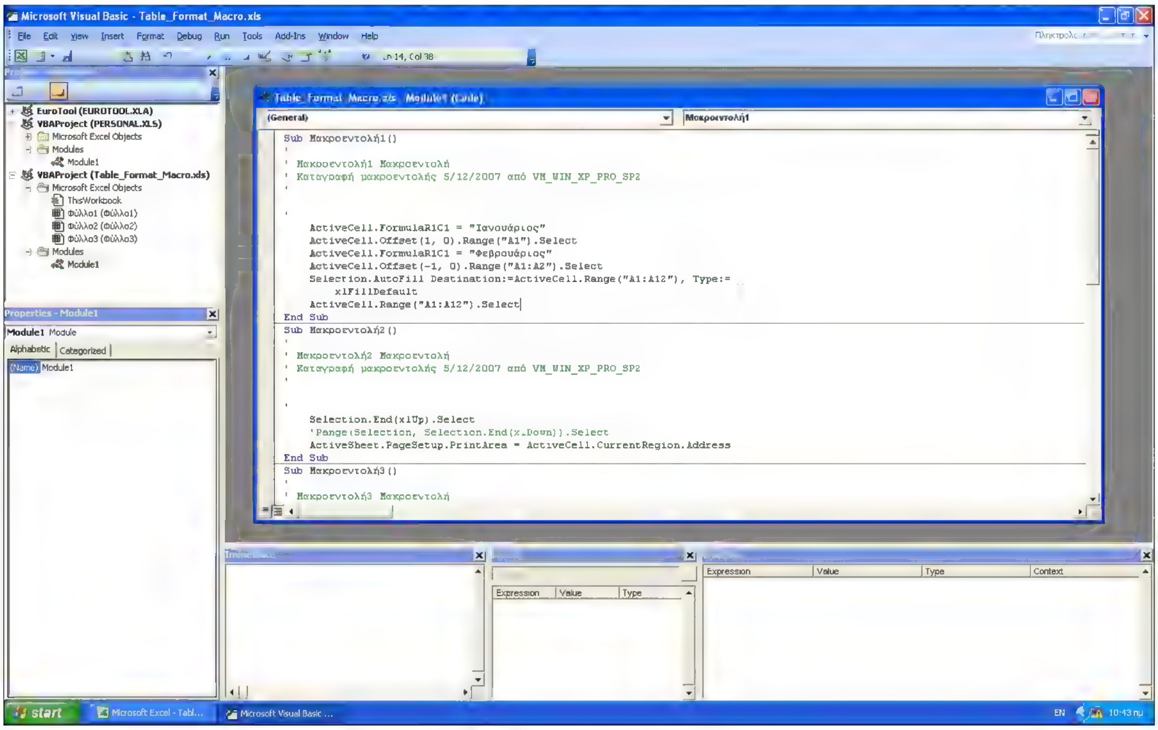
Ο Επεξεργαστής της Visual Basic

Τι υπάρχει πίσω από μία μακροεντολή

Όπως αναφέραμε ήδη, καθώς καταγράφεται μία μακροεντολή, οι εντολές που την απαρτίζουν μεταφράζονται σε κώδικα Visual Basic for Applications (VBA). Τον κώδικα VBA μπορούμε στη συνέχεια να τροποποιήσουμε.

Που βρίσκεται όμως αυτός ο κώδικας και πως μπορούμε να τον δούμε;

Ο κώδικας αποθηκεύεται σε ένα **VBA module**, και μπορούμε να τον δούμε μέσω του κατάλληλου εργαλείου που μας προσφέρει το Excel (όπως και οι υπόλοιπες εφαρμογές του Microsoft Office). Πρόκειται για τον **Επεξεργαστή Visual Basic**, τον οποίο ανοίγουμε από το μενού **Εργαλεία - Μακροεντολή – Επεξεργασία Visual Basic** ή με τη χρήση των πλήκτρων συντόμευσης **Alt+F11**.



Εικόνα 20 Ο επεξεργαστής Visual Basic

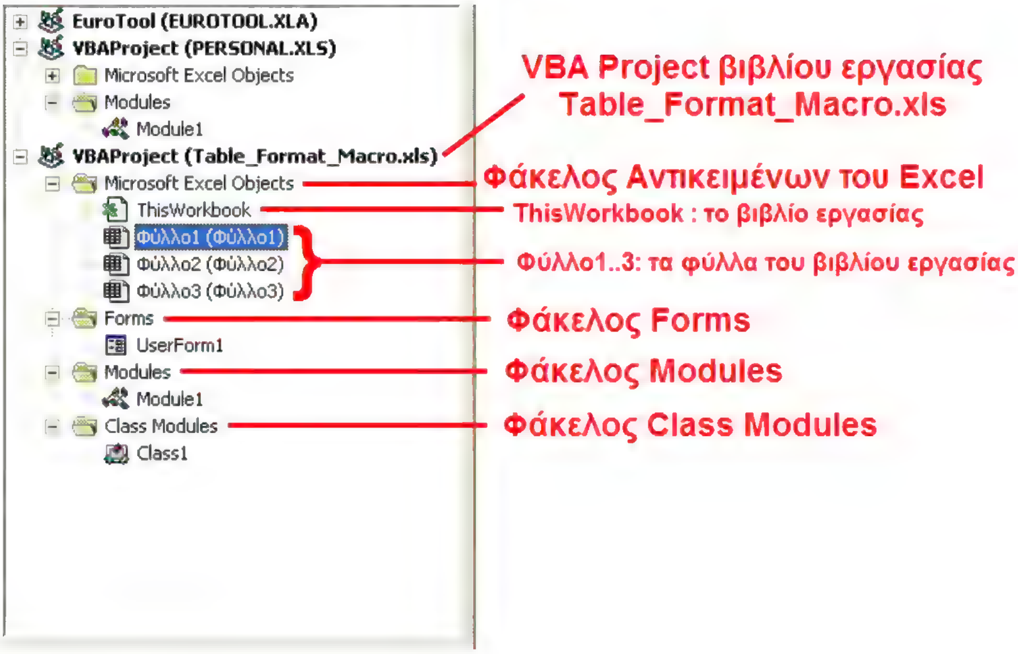
Κάθε μακροεντολή που καταγράφουμε αντιστοιχεί σε μία **διαδικασία (procedure)**. Ένα module μπορεί να περιέχει μία ή περισσότερες διαδικασίες. Η διαδικασία αντιστοιχεί στο χαμηλότερο επίπεδο που μπορεί να αναλυθεί ο κώδικας VBA.

```
Sub Μακροεντολή1()
'
' Μακροεντολή1 Μακροεντολή
' Καταγραφή Μακροεντολής 5/12/2007 από VM_WIN_XP_PRO_SP2
'

'
    ActiveCell.FormulaR1C1 = "Ιανουάριος"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Φεβρουάριος"
    ActiveCell.Offset(-1, 0).Range("A1:A2").Select
    Selection.AutoFill Destination:=ActiveCell.Range("A1:A12"), Type:= _
        xlFillDefault
    ActiveCell.Range("A1:A12").Select
End Sub
```

Εικόνα 21 Η διαδικασία της μακροεντολής Μακροεντολή1

Το σύνολο του κώδικα που υπάρχει σε ένα βιβλίο εργασίας αποτελεί ένα **VBA project**. Δηλαδή σε κάθε βιβλίο εργασίας αντιστοιχεί ένα VBA project.



Εικόνα 22 Τα αντικείμενα που μπορεί να περιέχει ένα VBA Project

Κάθε τέτοιο project περιέχει οπωσδήποτε το φάκελο Microsoft Excel Objects με τα παρακάτω αντικείμενα:

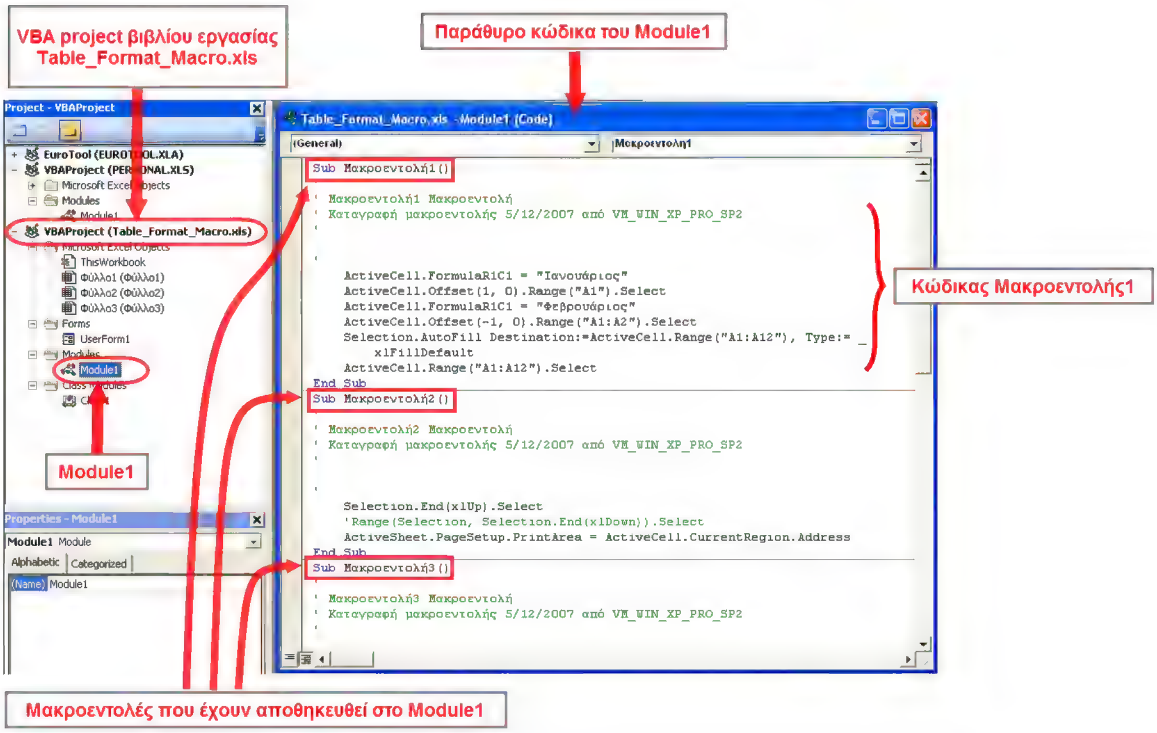
- **ThisWorkbook** που είναι το βιβλίο εργασίας
- **Φύλλο1 ... ΦύλλοΝ**, που αντιστοιχούν στα Ν φύλλα εργασίας που μπορεί να περιέχει το βιβλίο εργασίας

Κάθε ένα από τα αντικείμενα αυτά μπορεί να περιέχει κώδικα VBA.

Επίσης μπορεί να περιέχει τους παρακάτω φακέλους:

- **Modules**, με ένα κόμβο για κάθε Module (Module1, κλπ) , αν έχουν καταγραφεί μακροεντολές ή κώδικας VBA που γράψαμε απευθείας στον επεξεργαστή Visual Basic
- **Forms**, με ένα κόμβο για κάθε φόρμα χρήστη (UserForm1, κλπ)
- **Class Modules**, με ένα κόμβο για κάθε κλάση (Class1, κλπ) που έχει δημιουργηθεί

Τα ονόματα των ModuleX, UserFormX και ClassX δεν είναι δεσμευτικά και αν ο χρήστης επιθυμεί έχει τη δυνατότητα να τα αλλάξει.

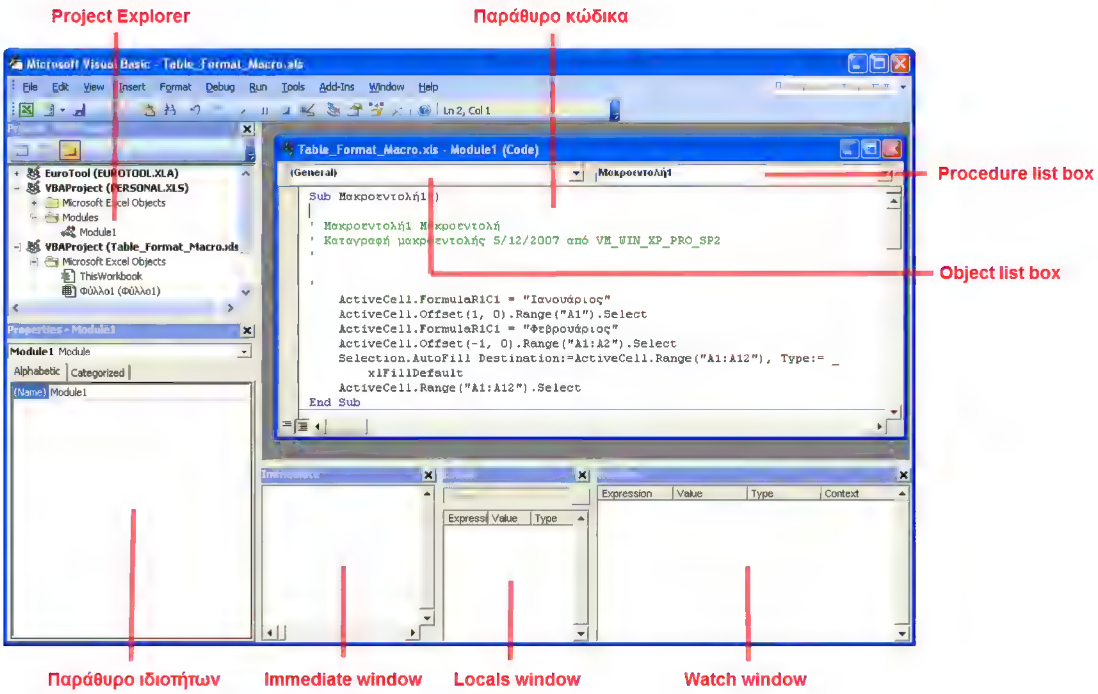


Εικόνα 23 Παράδειγμα ενός VBA Project με ένα module που περιέχει τρεις μακροεντολές

Τα βασικά στοιχεία του επεξεργαστή Visual Basic

Project Explorer

Περιέχει το σύνολο των ανοικτών projects (και κατά συνέπεια των ανοικτών βιβλίων εργασίας), των αντίστοιχων modules, των αντικειμένων και φορμών. Σε κάθε ένα από τα παραπάνω στοιχεία αντιστοιχεί ένας κόμβος. Το κάθε ένα από τα παραπάνω μπορεί να περιέχει κώδικα VBA.



Εικόνα 24 Τα βασικά στοιχεία του επεξεργαστή Visual Basic

Παράθυρο ιδιοτήτων (Properties window)

Εδώ εμφανίζονται οι ιδιότητες για το επιλεγμένο αντικείμενο στον project explorer. Μπορούμε να αλλάξουμε το όνομα και άλλες ιδιότητες κάθε στοιχείου που έχουμε επιλέξει στον project explorer.

Παράθυρο κώδικα (Code window)

Στο παράθυρο αυτό μπορούμε να δούμε, να τροποποιήσουμε ή να γράψουμε τον κώδικα που περιέχεται σε ένα module, ένα φύλλο εργασίας, ένα βιβλίο εργασίας, μία φόρμα χρήστη ή μία κλάση. Κάθε αντικείμενο στον Project Explorer έχει και το δικό του παράθυρο κώδικα, το οποίο είναι άδειο,

εκτός αν εμείς γράψουμε κώδικα VBA, ή καταγράψουμε μακροεντολή. Για να ανοίξει το παράθυρο κώδικα ενός αντικειμένου, κάνουμε διπλό κλικ με το ποντίκι πάνω στο αντικείμενο.

Στο επάνω μέρος του παραθύρου κώδικα υπάρχουν δύο list boxes.

Object list box

Εδώ εμφανίζονται αντικείμενα που σχετίζονται με το επιλεγμένο project.

Procedure list box

Εμφανίζονται οι διαδικασίες που μπορεί να περιέχει το επιλεγμένο αντικείμενο στο object list box.

Locals window

Εμφανίζονται οι τιμές των τοπικών μεταβλητών κατά τη βηματική εκτέλεση του κώδικα. Εμφανίζεται από την επιλογή του μενού **View – Locals window**.

Immediate window

Εδώ εμφανίζονται άμεσα τα αποτελέσματα που μπορεί να έχει η πληκτρολόγηση συγκεκριμένων εκφράσεων στο ίδιο παράθυρο, ή συγκεκριμένες εντολές κατά την εκτέλεση (βηματική ή όχι) του κώδικα. Εμφανίζεται από την επιλογή του μενού **View – Immediate window**, ή πατώντας **Ctrl+G**.

Watches window

Στο παράθυρο αυτό μπορούμε να επιλέξουμε συγκεκριμένες μεταβλητές, την τιμή των οποίων θέλουμε να παρακολουθούμε συνεχώς. Εμφανίζεται από την επιλογή του μενού **View – Watch Window.**

Τι παραπάνω μπορούμε να κάνουμε με τη VBA σε σχέση με τις μακροεντολές

Η χρήση της VBA στο Excel μας δίνει επιπλέον δυνατότητες σε σχέση με την απλή καταγραφή μακροεντολών. Για παράδειγμα:

- Δημιουργία νέων συναρτήσεων τις οποίες μπορούμε να χρησιμοποιούμε στους τύπους που εισάγουμε σε φύλλα εργασίας. Οι συναρτήσεις μας θα εμφανίζονται στο παράθυρο **Εισαγωγή Συνάρτησης**, στην κατηγορία **Συναρτήσεις Χρήστη**.
- Ανάπτυξη ολοκληρωμένων εφαρμογών, με πρόσθετα εικονίδια, μενού, κουμπιά, παράθυρα βοηθείας και άλλα, όπου ο χρήστης θα καθοδηγείται για τις ενέργειες του από την εφαρμογή.
- Δημιουργία δικών μας, εξειδικευμένων και προσαρμοσμένων στις ανάγκες μας **πρόσθετων** (**add-ins**). Επιπλέον μπορούμε να διανείμουμε τα πρόσθετα και σε άλλους χρήστες.

Τα συστατικά της VBA

Τα βασικά συστατικά της VBA είναι:

- **Κώδικας** (**code**): Μέσω του κώδικα της VBA, γενικά, πραγματοποιούμε λειτουργίες. Γράφουμε ή καταγράφουμε κώδικα, που αποθηκεύεται σε modules
- **Modules**: Τα VBA modules αποθηκεύονται στα βιβλία εργασίας του Excel. Η πρόσβαση σε αυτά παρέχεται μόνο μέσω του Επεξεργαστή VB
- **Διαδικασίες** (**procedures**): Είναι μονάδες κώδικα που εκτελούν κάποια, κατά βάση, αυτόνομη και απλή λειτουργία. Υπάρχουν δύο είδη διαδικασιών:

    - **Υπορουτίνες** (**Sub**): Είναι δυνατό, να δέχονται τιμές σε ορίσματα (χωρίς αυτό να είναι απαραίτητο) και εκτελούν μία ή περισσότερες ενέργειες. Οι μακροεντολές αποθηκεύονται σαν υπορουτίνες. Μπορούν να κληθούν από άλλες διαδικασίες. Το παρακάτω παράδειγμα είναι μία υπορουτίνα που ονομάζεται **Sub_demo_1**, εκτελεί ένα απλό πολλαπλασιασμό και καταχωρεί το αποτέλεσμα στο κελί A1.

```
Sub Sub_demo_1( )
   Answer = 2 * 5
   Cells(1, 1) = Answer
End Sub
```

    - **Συναρτήσεις** (**Functions**): Οι συναρτήσεις συνήθως εκτελούν κάποιους υπολογισμούς και επιστρέφουν πάντα μία τιμή. Ακόμα, δέχονται **ορίσματα** (**arguments**) που χρησιμοποιούν για την εκτέλεση των υπολογισμών, όπως και οι γνωστές συναρτήσεις του Excel. Καλούνται από άλλες διαδικασίες, ή μπορεί να εμφανίζονται ως συναρτήσεις χρήστη (**user functions**) στο Excel. Το παρακάτω παράδειγμα είναι μία συνάρτηση που ονομάζεται **Mult**, δέχεται τα ορίσματα **arg1** και **arg2**, εκτελεί ένα απλό πολλαπλασιασμό και επιστρέφει το γινόμενο.

```
Function Mult(arg1, arg2)
   Mult = arg1 * arg2
End Function
```

- **Αντικείμενα** (**objects**): Η VBA χειρίζεται αντικείμενα της εφαρμογής του Excel, όπως **Application, Workbook, Worksheet, Chart, Range** και άλλα. Τα αντικείμενα μπορεί να περιέχουν άλλα αντικείμενα. Για παράδειγμα, το αντικείμενο **Application** περιέχει το αντικείμενο **Workbook**.
- **Συλλογές** (**collections**): Είναι συλλογές από όμοια αντικείμενα. Για παράδειγμα τα φύλλα ενός βιβλίου εργασίας, που είναι τα αντικείμενα **Worksheet** απαρτίζουν τη συλλογή **Worksheets**. Οι συλλογές είναι και αυτές αντικείμενα.
- **Ιεραρχία αντικειμένων** (**object hierarchy**): Τα αντικείμενα δομούνται σε μία συγκεκριμένη ιεραρχία. Μπορούμε να αναφερθούμε σε ένα αντικείμενο ως εξής:

  Application.Workbooks("Βιβλίο1.xls").Worksheets("Φύλλο1")

- **Ιδιότητες αντικειμένων** (**object properties**): Τα αντικείμενα έχουν συγκεκριμένες ιδιότητες, οι οποίες έχουν κάποιες τιμές. Για παράδειγμα το αντικείμενο **Range**, που αντιστοιχεί σε ένα κελί ή περιοχή κελιών, έχει τις ιδιότητες **Value** και **Name.** Η πρώτη ιδιότητααφορά στην τιμή του κελιού (ή της περιοχής κελιών) ενώ η δεύτερη στο όνομα του κελιού (ή της περιοχής κελιών). Μπορούμε να αναφερθούμε σε μία ιδιότητα ενός αντικειμένου, ή να της δώσουμε μία συγκεκριμένη τιμή ως εξής:

  Worksheets("Φύλλο1").Range("A1").Value = 100

- **Μέθοδοι αντικειμένων** (**object methods**): Μπορούμε να χειριστούμε τα αντικείμενα μέσω συγκεκριμένων μεθόδων, δηλαδή ενεργειών που εκτελούνται με αυτά. Για παράδειγμα το αντικείμενο Range έχει την ιδιότητα **ClearContents**, η οποία διαγράφει τα περιεχόμενα του κελιού ή της περιοχής κελιών. Εκτελούμε μία μέθοδο ενός αντικειμένου ως εξής:

  Range("A1").ClearContents

  Μία μέθοδος μπορεί να έχει μία ή περισσότερες παραμέτρους. Στην περίπτωση όπου η μέθοδος έχει μόνο μία παράμετρο και θέλουμε της δώσουμε μία τιμή, χρησιμοποιούμε την παρακάτω σύνταξη:

  objectname.methodname parametername=value

  όπου **objectname** είναι το όνομα του αντικειμένου, **methodname** είναι το όνομα της μεθόδου, **parametername** το όνομα της παραμέτρου και **value** η τιμή που της δίδουμε. Για παράδειγμα, η παρακάτω πρόταση κώδικα

  Range("B3:B4").Delete Shift:=xlShiftUp

  Αναφέρεται στη μέθοδο **Delete** του αντικειμένου **Range** , η οποία έχει ως παράμετρο την **Shift** που της αποδίδουμε την τιμή **xlShiftUp**

Αν μία μέθοδος έχει περισσότερες από μία παραμέτρους τότε μπορούμε να δώσουμε τιμή σε κάποιες ή όλες από αυτές. Αν θέλουμε να δώσουμε τιμή σε μερικές μόνο, χρησιμοποιούμε την παρακάτω σύνταξη:

Workbooks.Open(Filename:="test.xls", ReadOnly:=True)

Εδώ η μέθοδος Open έχει περισσότερες παραμέτρους (FileName, UpdateLinks, ReadOnly, Format, Password, WriteResPassword, IgnoreReadOnlyRecommended, Origin, Delimiter, Editable, Notify, Converter, AddToMru, Local, CorruptLoad), εμείς όμως θέλουμε να δώσουμε τιμή μόνο σε δυο από αυτές και συγκεκριμένα στις Filename και ReadOnly.

- **Συμβάντα** (**events**): Η VBA μας δίνει τη δυνατότητα να εκτελούμε κώδικα, όταν συμβαίνουν συγκεκριμένα γεγονότα στην εφαρμογή του Excel. Τέτοια παραδείγματα είναι α)το άνοιγμα ενός βιβλίου εργασίας, β) η αλλαγή ενός φύλλου εργασίας, ή γ) το κλείσιμο ενός βιβλίου εργασίας. Ο κώδικας πρέπει να βρίσκεται σε υπορουτίνες με συγκεκριμένα ονόματα, όπως BeforeClose, για να εκτελείται όταν πρέπει. Τα συμβάντα συνδέονται με τα αντικείμενα, και κάθε είδος αντικειμένου υποστηρίζει διαφορετικά συμβάντα.
- **Μεταβλητές** (**variables**): Η μεταβλητές είναι οντότητες, θέσεις στη μνήμη του υπολογιστή, που χαρακτηρίζονται από ένα μοναδικό όνομα και χρησιμοποιούνται για να αποθηκεύουμε πληροφορίες. Στο προηγούμενο παράδειγμα της υπορουτίνας **Sub_demo_1** η **Answer** είναι μία μεταβλητή.

Τα modules

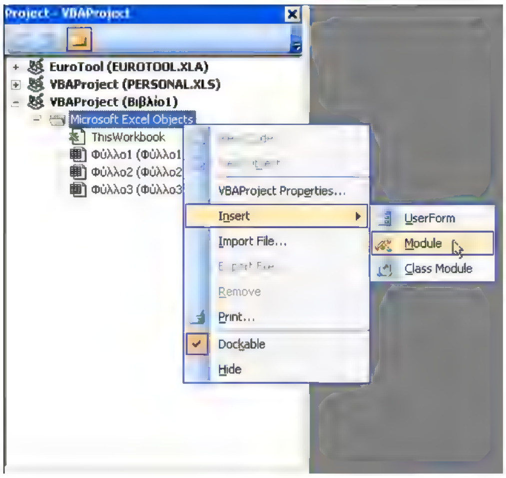
Γενικά τα modules μπορεί να περιέχουν κώδικα VBA στις παρακάτω μορφές:

- **Δηλώσεις** (**declarations**), που χρησιμοποιούμε στη VBA, για παράδειγμα για να ορίσουμε τον τύπο των δεδομένων που αποθηκεύει μία μεταβλητή, ή να ορίσουμε κάποιες παραμέτρους για το module.
- **Υπορουτίνες (Sub)**, δηλαδή κώδικα VBA που εκτελεί μία ενέργεια.
- **Συναρτήσεις (Function)**, δηλαδή κώδικα VBA που εκτελεί υπολογισμούς και επιστρέφει μία τιμή.

Ένα module μπορεί να περιέχει οποιοδήποτε αριθμό και συνδυασμό δηλώσεων, υπορουτινών και συναρτήσεων. Η οργάνωση του κώδικα VBA που γράφουμε σε υπορουτίνες, συναρτήσεις ή και modules είναι θέμα προσωπικής επιλογής.

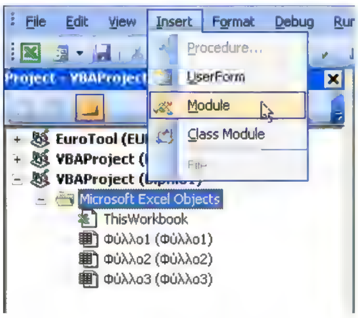
Πως προσθέτουμε, διαγράφουμε ή μετονομάζουμε ένα Module σε ένα VBA Project

Το VBA Project ενός βιβλίου εργασίας δεν έχει κανένα module, εκτός και αν έχουμε καταγράψει κάποια μακροεντολή. Για να εισάγουμε ένα module επιλέγουμε το VBA project στον Project Explorer και στη συνέχεια πατάμε το δεξί κουμπί του ποντικιού και στο μενού που εμφανίζεται επιλέγουμε **Insert – Module**



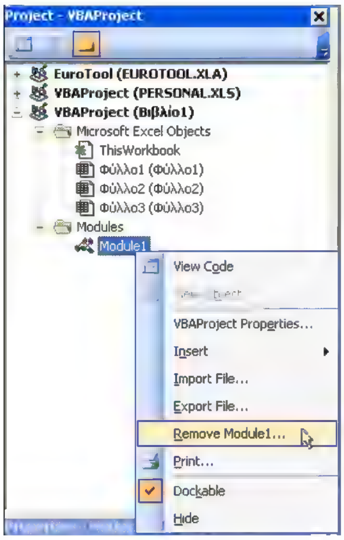
Εικόνα 25 Εισαγωγή νέου module σε ένα VBA Project

ή από τη γραμμή μενού επιλέγουμε **Insert – Module**



Εικόνα 26 Εισαγωγή νέου module σε ένα VBA Project από τη γραμμή μενού

Για να διαγράψουμε ένα module πατάμε το δεξί κουμπί του ποντικιού πάνω από το συγκεκριμένο module, στο μενού που εμφανίζεται επιλέγουμε **Remove ModuleX** και απαντάμε στην ερώτηση που μας γίνεται ανάλογα με το αν θέλουμε να διαγράψουμε το module ή όχι.



Εικόνα 27 Διαγραφή ενός module από ένα VBA Project

Για να μετονομάσουμε ένα module, το επιλέγουμε στον Project Explorer και στο παράθυρο ιδιοτήτων εισάγουμε το νέο όνομα στο πεδίο **Name**.



Εικόνα 28 Μετονομασία module

Υπορουτίνες

Όπως έχουμε αναφέρει, οι υπορουτίνες (subroutines ή συνηθέστερα sub) είναι διαδικασίες που εκτελούν μία ή περισσότερες ενέργειες. Οι υπορουτίνες μπορούν να κληθούν από άλλες διαδικασίες και οι μακροεντολές αποθηκεύονται σε μορφή υπορουτινών.

Δήλωση μίας υπορουτίνας

Κατά το στάδιο της δημιουργίας της, μία υπορουτίνα πρέπει να δηλωθεί με ένα συγκεκριμένο τρόπο. Ανοίγουμε το παράθυρο κώδικα του module στο οποίο θέλουμε να δημιουργήσουμε την υπορουτίνα και πληκτρολογούμε:

1. Τη λέξη **Sub**
2. Το όνομα που θέλουμε να έχει και
3. τις παρενθέσεις ()

Μόλις πληκτρολογήσουμε τις παρενθέσεις και πατήσουμε Enter, αυτόματα προστίθεται η εντολή **End Sub**, που δηλώνει το τέλος του κώδικα που περιέχει η συγκεκριμένη υπορουτίνα. Μετά την γραμμή που δηλώσαμε την υπορουτίνα και πριν την εντολή **End Sub** εισάγουμε τον κώδικα που επιθυμούμε. Η παρακάτω υπορουτίνα, ονομάζεται **Add_Cells** και έχει σαν αποτέλεσμα να προσθέτει τα περιεχόμενα των κελιών A1 και B1 και καταχωρεί το αποτέλεσμα στο κελί C1.

```
Sub Add_Cells()
        a = Cells(1,1)
        b = Cells(1,2)
        c = a + b
        Cells(1,3) =c
End Sub
```

Η υπορουτίνα **Add_Cells** αποτελείται από τρία τμήματα.

1. Η γραμμή δήλωσης της υπορουτίνας:

```
Sub Add_Cells()
```

2. Οι γραμμές που περιέχουν τον κώδικα της υπορουτίνας:

```
a = Cells(1,1)
b = Cells(1,2)
c = a + b
Cells(1,3) =c
```

3. Η γραμμή που δηλώνει το τέλος της υπορουτίνας:

```
End Sub
```

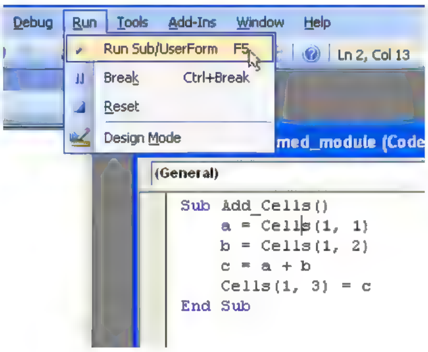
Σε κάθε γραμμή κώδικα αντιστοιχεί συνήθως μία εντολή. Αν η εντολή έχει μεγάλο μήκος και θέλουμε να την χωρίσουμε σε δύο ή περισσότερες γραμμές πρέπει να πληκτρολογήσουμε στο τέλος κάθε γραμμής, εκτός της τελευταίας, τους χαρακτήρες **space** και **underscore**, δηλαδή " _". Για παράδειγμα:

```
Application.Workbooks("Βιβλίο1.xls").Worksheets("Φύλλο1") _
Range("A1").Value = 100
```

Είναι συνήθης προγραμματιστική πρακτική να προσθέτουμε εσοχές στον κώδικα, για να είναι ευανάγνωστος.

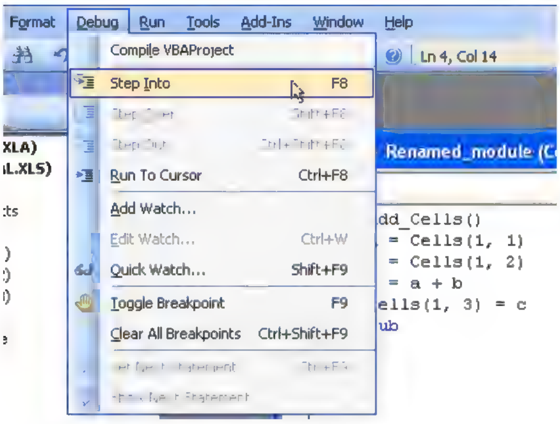
Εκτέλεση υπορουτινών

Για να εκτελέσουμε μία υπορουτίνα κάνουμε κλικ με το ποντίκι οπουδήποτε εντός του σώματός της (στο παράθυρο του κώδικα) και από τη γραμμή μενού επιλέγουμε **Run – Run Sub/UserForm** ή πατάμε **F5**. Η μέθοδος αυτή εκτελεί όλες τις εντολές που περιλαμβάνει ο κώδικας της υπορουτίνας μέχρι την εντολή End Sub, όπου και τερματίζεται η εκτέλεση.



Εικόνα 29 Εκτέλεση υπορουτίνας

Αν θέλουμε να εκτελέσουμε μία προς μία τις εντολές που περιέχει ο κώδικας της υπορουτίνας κάνουμε κλικ με το ποντίκι οπουδήποτε εντός της υπορουτίνας (στο παράθυρο κώδικα) και από τη γραμμή μενού επιλέγουμε **Debug – Step Into** ή πατάμε **F8.**



Εικόνα 30 Βηματική εκτέλεση κώδικα

Κλήση υπορουτινών από άλλες διαδικασίες

Είναι δυνατό, μία υπορουτίνα να καλεί άλλες υπορουτίνες. Η VBA μας παρέχει τρεις μεθόδους για την κλήση υπορουτινών:

1. Εισάγουμε το όνομα της υπορουτίνας, και τυχόν ορίσματα που έχει, χωρίζοντας τα με κόμμα:

```
Sub Master_sub()
   a = 1
   b = 2
   Called_Sub a, b
End Sub
```

```
Sub Called_Sub(arg1, arg2)
   Cells(1, 1) = arg1 + arg2
End Sub
```

Στο παραπάνω παράδειγμα, ο κώδικας της υπορουτίνας Master_sub, αποδίδει αρχικά τιμές στις μεταβλητές a και b, και στη συνέχεια καλεί την υπορουτίνα Called_Sub και της «περνάει» ως ορίσματα τις μεταβλητές a και b. Στο σημείο αυτό η VBA αναστέλλει την εκτέλεση του κώδικα της υπορουτίνας Master_sub, μεταβαίνει στην υπορουτίνα Called_Sub και εκτελεί τον κώδικα της. Όταν τελειώσει η εκτέλεση αυτού του κώδικα, η VBA επανέρχεται στην επόμενη γραμμή του κώδικα της υπορουτίνας Sub_Master από όπου είχε φύγει, και συνεχίζει από εκεί την εκτέλεση.

2. Χρησιμοποιώντας την εντολή Call, όπως παρακάτω:

```
Sub Master_sub()
   a = 1
   b = 2
   Call Called_Sub(a, b)
End Sub
```

```
Sub Called_Sub(arg1, arg2)
   Cells(1, 1) = arg1 + arg2
End Sub
```

3. Με τη βοήθεια της μεθόδου Run του αντικειμένου Application:

```
Sub Master_sub()
   a = 1
   b = 2
   Application.Run "Called_Sub", a, b
End Sub
```

```
Sub Called_Sub(arg1, arg2)
   Cells(1, 1) = arg1 + arg2
End Sub
```

Το αποτέλεσμα είναι το ίδιο και στα τρία παραδείγματα, δηλαδή: προστίθενται οι αριθμοί 1 και 2 και το αποτέλεσμα εισάγεται στο κελί A1 του ενεργού φύλλου εργασίας.

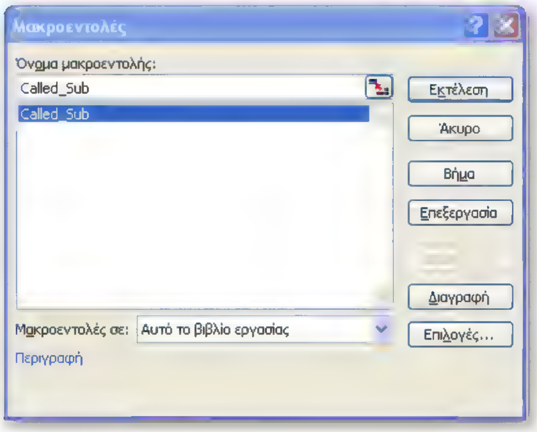
Κλήση υπορουτινών από το περιβάλλον του Excel

Όπως είδαμε, ο κώδικας μίας μακροεντολής περιέχεται μέσα σε μία υπορουτίνα, και συνεπώς κάθε υπορουτίνα μπορεί να εμφανίζεται ως μακροεντολή στο παράθυρο **Μακροεντολές** από το μενού **Εργαλεία – Μακροεντολή**. Για να εμφανίζεται όμως η υπορουτίνα δεν πρέπει να έχει κανένα όρισμα. Έτσι τροποποιούμε την προηγούμενη ρουτίνα:

```
Sub Called_Sub
```

```
    arg1 = 1
    arg2 = 2
    Cells(1, 1) = arg1 + arg2
End Sub
```

Μόλις καλέσουμε την υπορουτίνα από το παράθυρο Μακροεντολές θα εμφανισθεί στο κελί A1 η τιμή 3.



Εικόνα 31 Κλήση της υπορουτίνας Called_Sub από το παράθυρο Μακροεντολές

Μεταβλητές

Μία μεταβλητή (variable) δεν είναι παρά ένα χώρος αποθήκευσης στη μνήμη του υπολογιστή που χαρακτηρίζεται από ένα όνομα. Στη μεταβλητή αποθηκεύονται δεδομένα, οι τιμές των οποίων μπορεί να μεταβληθούν κατά την εκτέλεση του κώδικα. Γενικά είναι καλή συνήθεια να χρησιμοποιούμε ονόματα που υποδηλώνουν τη χρήση των μεταβλητών στο κώδικα (δηλαδή τι κάνουμε με αυτές τις μεταβλητές). Υπάρχουν όμως μερικοί κανόνες που αφορούν στη δημιουργία αποδεκτών ονομάτων μεταβλητών, στη VBA. Αυτοί είναι:

- Μπορούμε να χρησιμοποιήσουμε χαρακτήρες όπως γράμματα, αριθμούς και μερικά σύμβολα. Ο πρώτος χαρακτήρας όμως πρέπει να είναι γράμμα και ο μέγιστος αριθμός χαρακτήρων 256
- Το όνομα μίας μεταβλητής δεν μπορεί να είναι μία δεσμευμένη έκφραση της VBA
- Δεν υπάρχει διάκριση μεταξύ πεζών και κεφαλαίων χαρακτήρων
- Δεν επιτρέπεται η χρήση των παρακάτω χαρακτήρων
    - τελεία "."
    - μονό διάστημα " "
    - #, $, %, & και !

Επιτρέπεται όμως η χρήση του χαρακτήρα "_" (underscore)

Δήλωση μεταβλητών

Ο καθορισμός του τύπου μίας μεταβλητής, κατά το στάδιο της δημιουργίας της αποτελεί καλή προγραμματιστική πρακτική μια και μας προφυλάσσει από σφάλματα καταχωρήσεων κ.λ.π. Αυτό γίνεται με την παρακάτω σύνταξη:

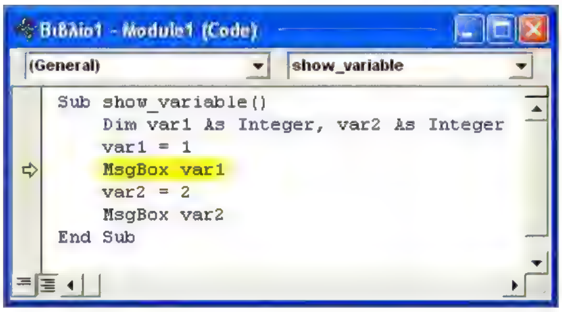
Dim variablename **as** datatype

Η λέξη **variablename** αναφέρεται στο όνομα της μεταβλητής ενώ η λέξη **datatype** στον τύπο δεδομένων.

Στο παρακάτω παράδειγμα, δηλώνουμε μία μεταβλητή τύπου integer, αποδίδουμε σε αυτή την τιμή 100 και την εμφανίζουμε με τη βοήθεια της συνάρτησης **MsgBox**:

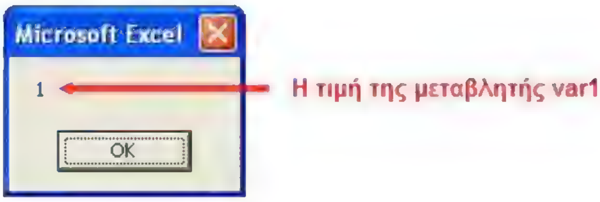
```
Sub show_variable()
    Dim var1 As Integer, var2 As Integer
    var1 = 1
    MsgBox var1
    var2 = 2
    MsgBox var2
End Sub
```

Η συνάρτηση **MsgBox** μπορεί να εμφανίζει ένα **μήνυμα** (**message**) σε ένα **πλαίσιο διαλόγου** (**dialog box**), και επίσης να δέχεται την απόκριση του χρήστη όταν αυτός πατάει κουμπιά που εμφανίζονται στο παράθυρο του μηνύματος.



Εικόνα 32 Η συνάρτηση MsgBox

Μόλις η εκτέλεση του κώδικα φτάσει στη γραμμή όπου υπάρχει η συνάρτηση MsgBox παρουσιάζεται το πλαίσιο διαλόγου, όπου εμφανίζεται η τιμή της μεταβλητής var1 (1):



Εικόνα 33 Το παράθυρο όπου εμφανίζεται η τιμή της μεταβλητής var1

Μπορούμε να αποδώσουμε τιμή απευθείας σε μία μεταβλητή, χωρίς προηγουμένως να τη δηλώσουμε και να ορίσουμε τον τύπο των δεδομένων που αποθηκεύονται σε αυτή. Παρακάτω βλέπουμε τροποποιημένο το προηγούμενο παράδειγμα:

```
Sub show_variable()
   var1 = 1
   MsgBox var1
   var2 = 2
   MsgBox var2
End Sub
```

Στη περίπτωση αυτή κάθε μεταβλητή δημιουργείται τη στιγμή που της αποδίδεται μία τιμή. Έτσι όμως είναι εύκολο να προκύψουν λάθη, όπως για παράδειγμα αν γράψουμε λάθος το όνομα μίας μεταβλητής, όπως παρακάτω (va1 αντί var1):

```
Sub show_variable()
   var1 = 1
   MsgBox var1
   var2 = 2
   MsgBox var2
   var1 = 3
   MsgBox va1
End Sub
```

Εδώ η μεταβλητή va1 δεν έχει τιμή, και επομένως το πλαίσιο διαλόγου που εμφανίζει η **MsgBox** δεν έχει καμία πληροφορία.

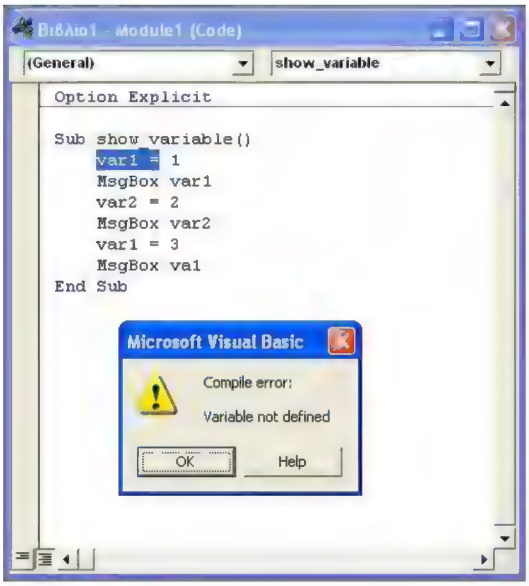


Εικόνα 34 Στο παράθυρο μηνύματος δεν εμφανίζεται τιμή για τη μεταβλητή va1

Προκειμένου να αποφύγουμε παρόμοια προβλήματα, μπορούμε να ενημερώσουμε τη VBA ότι κάθε μεταβλητή θα πρέπει να έχει δηλωθεί πριν χρησιμοποιηθεί, εισάγοντας την παρακάτω δήλωση στην αρχή του module, πριν από τον κώδικα οποιασδήποτε υπορουτίνας ή συνάρτησης:

Option Explicit

Μόλις προσπαθήσουμε να εκτελέσουμε τον κώδικα μίας υπορουτίνας όπου οι μεταβλητές δεν έχουν δηλωθεί εμφανίζεται ένα μήνυμα σφάλματος, όπως φαίνεται στην παρακάτω εικόνα, και δεν είναι δυνατή η συνέχιση της εκτέλεσης του κώδικα:



Εικόνα 35 Το μήνυμα σφάλματος που μας πληροφορεί ότι δεν έχουν δηλωθεί μεταβλητές που υπάρχουν στην υπορουτίνα που εκτελούμε

Εύρος (scope) μεταβλητών

Ανάλογα με το σημείο του κώδικα και το τρόπο δήλωσης της μία μεταβλητή μπορεί να είναι ορατή:

1. μόνο μέσα σε μία διαδικασία
2. μόνο σε ένα module
3. σε όλα τα modules που περιέχει ένα βιβλίο εργασίας.

Εύρος	**Δήλωση**	**Σημείο δήλωσης**
Εντός μίας διαδικασίας	Dim ή Static	Εντός της διαδικασίας
Εντός ενός module	Dim ή Private	Εντός του module και πριν από την πρώτη διαδικασία
Σε όλα τα modules	Public	Εντός ενός module και πριν από την πρώτη διαδικασία

Πίνακας 1 Εύρος μεταβλητών

Τοπικές μεταβλητές (local variables)

Οι τοπικές μεαβλητές δηλώνονται μέσα σε μία διαδικασία (οπουδήποτε μεταξύ των εκφράσεων **Sub** και **End Sub**). Η μεταβλητή παύει να υπάρχει όταν τελειώσει η εκτέλεση της διαδικασίας. Αν μία μεταβλητή δηλωθεί με τοπικό εύρος, είναι δυνατόν να υπάρχουν άλλες τοπικές μεταβλητές σε άλλες διαδικασίες με το ίδιο όνομα, κάθε μία μοναδική εντός της διαδικασίας που έχει δηλωθεί.

Μεταβλητές ορατές σε όλες τις διαδικασίες ενός module (modulewide variables)

Όταν επιθυμούμε μία μεταβλητή να είναι ορατή σε όλες τις διαδικασίες ενός module, τη δηλώνουμε πριν την πρώτη διαδικασία, όπως παρακάτω:

```
Dim modulewidevariable As Integer

Sub Sub1()
    ...
End Sub

Sub Sub2()
    ...
End Sub
```

Η τιμή της μεταβλητής δεν αλλάζει όταν τελειώσει η εκτέλεση μίας διαδικασίας. Εξαίρεση αποτελεί η περίπτωση όπου η εκτέλεση του κώδικα μίας διαδικασίας σταματήσει με την εντολή **End**.

Δημόσιες μεταβλητές (public variables)

Αν μία μεταβλητεί δηλωθεί με την έκφραση **Public** αντί **Dim**, πριν την πρώτη διαδικασία ενός module, τότε αυτή είναι ορατή σε όλα τα modules του βιβλίου εργασίας. Σε αυτή την περίπτωση η μεταβλητή είναι προσβάσιμη από όλες τις διαδικασίες ανεξάρτητα από το module που βρίσκονται.

Στατικές μεταβλητές (static variables)

Οι στατικές μαεταβλητές δηλώνονται εντός μίας διαδικασίας με την έκφραση **Static**, αλλά δεν χάνουν την τιμή τους όταν τελειώσει η εκτέλεση του κώδικα της διαδικασίας.

Τύποι δεδομένων

Η VBA χειρίζεται αυτόματα και κατ΄εκτίμηση τα δεδομένα μίας μεταβλητής αν η μεταβλητή αυτή δεν έχει δηλωθεί προηγουμένως, όσον αφορά τον τύπο της. Με τον τρόπο αυτό δεν γίνεται αποδοτική διαχείριση της μνήμης του υπολογιστή και ο κώδικας εκτελείται σχετικά αργά.

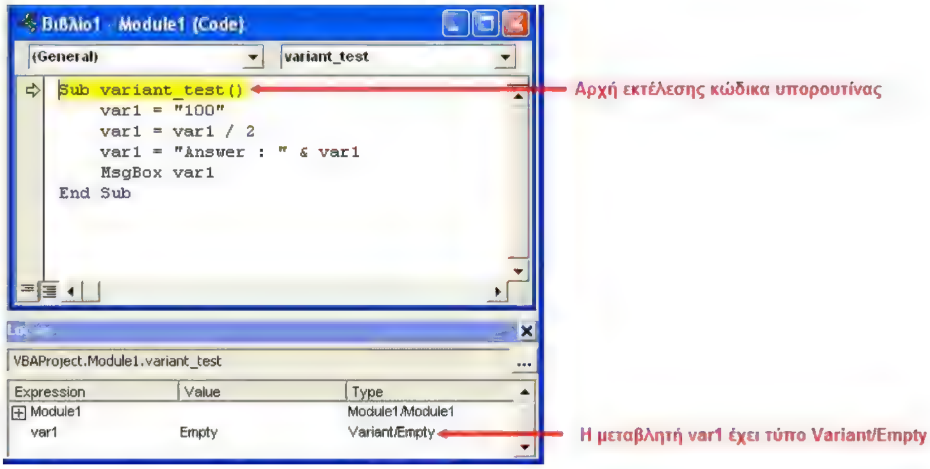
Στον Πίνακας 2 παρουσιάζονται οι τύποι δεδομένων που υποστηρίζει η VBA.

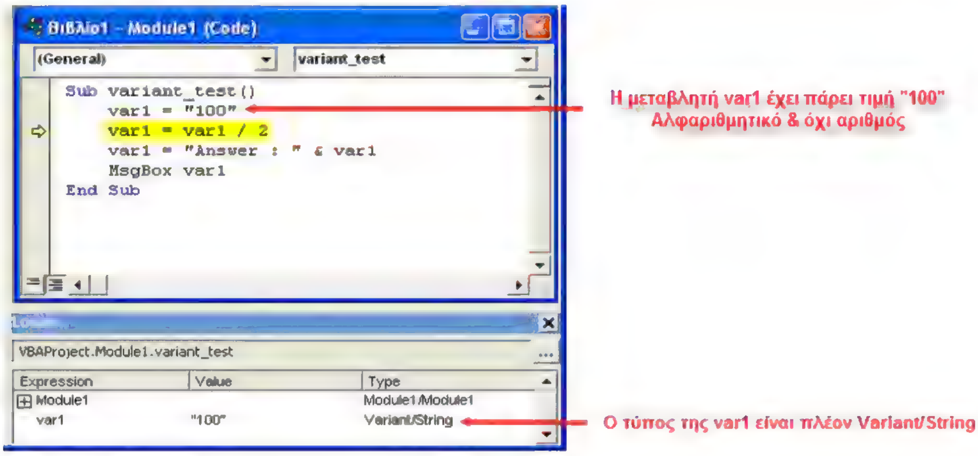
Τύπος δεδομένων	**Μέγεθος (Bytes)**	**Εύρος τιμών**
Byte	1	0 έως 255
Boolean	2	True ή False
Integer	2	-32.768 έως 32.767
Long	4	-2.147.483.648 έως 2.147.483.647
Single	4	-3,402823e^{38} έως -1,401298e^{-45} (αρνητικοί αριθμοί) και έως 1,401298e^{-45} έως 3,402823e^{38} (θετικοί αριθμοί)
Double	8	-1,79769313486232e^{308} έως -4,94065645841247e^{-324} (αρνητικοί αριθμοί) και έως 4,94065645841247e^{-324} έως 1,79769313486232e^{308} (θετικοί αριθμοί)
Currency	8	-922.337.203.685.477.5808 έως 922.337.203.685.477.5807
Decimal	14	+/- 79.228.162.514.264.337.593.543.950.335 χωρίς δεκαδικό μέρος, και +/- 7,9228162514264337593543950335 με 28 δεκαδικά ψηφία
Date	8	1^{η} Ιανουαρίου 100 έως 31^{η} Δεκεμβρίου 9999
Object	4	Οποιαδήποτε αναφορά σε αντικείμενο του Excel
String (fixed length)	Μέγεθος αλφαριθμητικού	1 μέχρι περίπου 64.500
String (variable length)	10 + μέγεθος αλφαριθμητικού	0 μέχρι περίπου 2.000.000
Variant (numbers)	16	Οποιαδήποτε αριθμητική τιμή εντός του εύρους τύπου double
Variant (characters)	22 + μέγεθος αλφαριθμητικού	0 μέχρι περίπου 2.000.000
User defined	Εξαρτώμενο	Εξαρτώμενο από τον τύπο

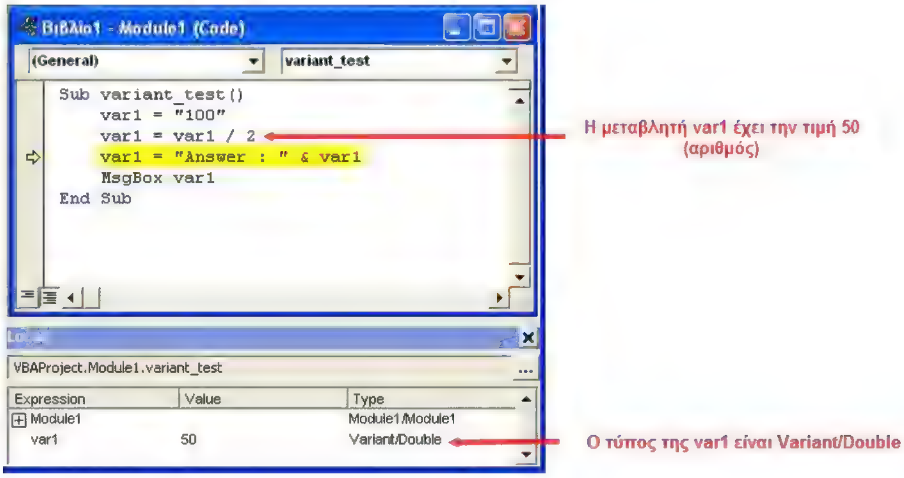
Πίνακας 2 Τύποι δεδομένων

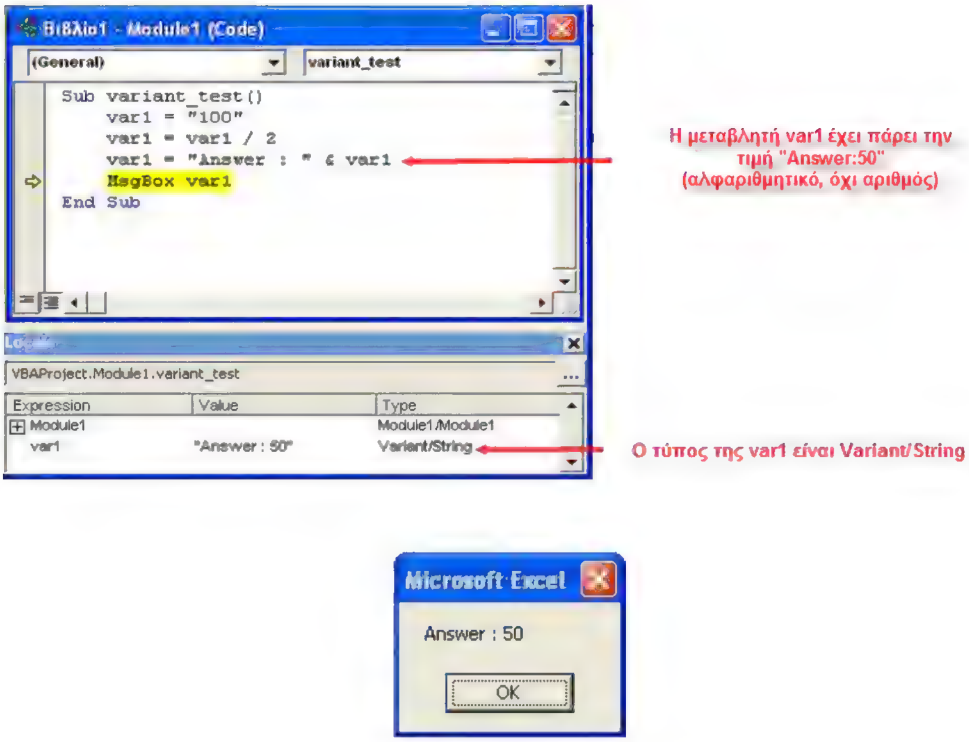
Αν δεν δηλώσουμε μία μεταβλητή, τότε το Excel χρησιμοποιεί για τη μεταβλητή αυτή τον τύπο δεδομένων variant. Είναι ένα είδος μεταβλητής που μπορεί να λειτουργήσει ως υποδοχέας διαφόρων

ειδών δεδομένων. Έτσι, αν στον κώδικα αποδίδουμε στη μεταβλητή τιμές διαφόρων τύπων δεδομένων, τότε μεταβάλλεται και ο τύπος της μεταβλητής ανάλογα κάθε φορά σε τρόπον ώστε να μπορεί να δεχθεί τη νέα τιμή, όπως φαίνεται στο επόμενο παράδειγμα:









Εικόνα 36 Η μεταβολή της τιμής και του τύπου μίας μεταβλητής που δέν έχει δηλωθεί

Πίνακες (arrays)

Οι **Πίνακες** (**arrays**) είναι ένα σύνολο στοιχείων του ίδιου τύπου, και χαρακτηρίζονται από ένα κοινό όνομα. Αναφερόμαστε σε ένα στοιχείο ενός πίνακα χρησιμοποιώντας το όνομα και έναν ή περισσότερους δείκτες (αριθμούς). Η δήλωση ενός πίνακα μίας διάστασης γίνεται ως εξής:

Dim VarArr(0 To 9) As String

ή

Dim VarArr(9) As String

Η μικρότερη τιμή του δείκτη είναι το 0 και όχι το 1. Επομένως ο πίνακας **VarArr** περιέχει 10 στοιχεία. Αποδίδουμε τιμή σε ένα στοιχείο του πίνακα ως εξής:

VarArr(0) = "Alfa"

Οι πίνακες έχουν δύο όρια, το κάτω και το άνω. Το προκαθορισμένο κάτω όριο είναι το 0 και το άνω όριο είναι αυτό που θα ορίσουμε κατά τη δήλωση του πίνακα. Αν επιθυμούμε το κάτω όριο να είναι το 1 και όχι το 0, αρκεί να εισάγουμε στην αρχή κάθε module (δηλαδή πριν από οποιαδήποτε διαδικασία) την παρακάτω έκφραση:

Option Base 1

Έχουμε επίσης τη δυνατότητα να ορίσουμε οποιοδήποτε κάτω όριο επιθυμούμε, κατά τη δήλωση του πίνακα, ως εξής:

Dim VarArr(5 To 9) As String

Σε αυτή την περίπρωση ο πίνακας έχει κάτω όριο το 5 και άνω το 9.

Οι πίνακες μπορούν να έχουν παραπάνω από μία διαστάσεις, όπως στα προηγούμενα παραδείγματα. Για τη δήλωση ενός πίνακα με περισότερες από μία διαστάσεις, διαχωρίζουμε τα όρια ανά διάσταση με κόμματα. Ο πίνακας **MulArr** έχει δύο διαστάσεις:

Dim MulArr(9, 19) as Integer

Μπορούμε να προσομοιάσουμε ένα πίνακα δύο διαστάσεων με ένα φύλλο εργασίας του Excel. Η μία διάσταση μπορεί να αντιστοιχεί στις γραμμές και η άλλη στις στήλες του φύλλου. Αν ο πίνακας έχει τρείς διαστάσεις, θα μπορούσαμε να αντιστοιχίσουμε την τρίτη στα φύλλα εργασίας ενός βιβλίου.

Αν δεν δηλώσουμε τον πλήθος των στοιχείων ενός πίνακα, τότε η δέσμευση μνήμης για να επιτευχθει η αποθήκευση γίνεται με **δυναμικό τρόπο**. Δηλαδή

Dim VarArr() As String

Πριν τη χρήση πρέπει να χρησιμοποιήσουμε την έκφραση **Redim** για να δηλώσουμε το πλήθος των στοιχείων. (ή **Redim Preserve** αν θέλουμε να διατηρηθούν οι τιμές των στοιχείων της).

Σταθερές

Στην περίπτωση που απαιτείται να έχουμε ένα χώρο αποθήκευσης που χαρακηρίζεται από ένα όνομα, αλλά η τιμή που αποθηκεύεται σ΄αυτόν δε μεταβάλλεται, χρησιμοποιούμε τις **σταθερές** (**constants**). Οι σταθερές δηλώνονται με παρόποιο τρόπο με τις μεταβλητές:

Const NumOfYears As Integer = 10

Αν δεν δηλώσουμε τον τύπο δεδομένων, η VBA τον καθορίζει βάσει της τιμής που αποδίδουμε στη σταθερά. Στο επόμενο παράδειγμα η σταθερά **Percentage** είναι τύπου **double** και η **NumOfOrders** τύπου **integer**:

Const Percentage = 0,001, NumOfOrders = 10

Οι σταθερές έχουν και αυτές συγκεκριμένο εύρος, όπως και οι μεταβλητές. Αν θέλουμε να δηλώσουμε μία σταθερά ώστε να είναι ορατή σε όλα τα modules, το κάνουμε ως εξής:

Public Const FullName As String = "Nick Gravenitis"

Εργασία με κελιά και περιοχές κελιών

Το αντικείμενο Range

Το αντικείμενο **Range** χρησιμοποιείται όταν θέλουμε να αναφερθούμε σε μία περιοχή κελιών. Η περιοχή μπορεί να είναι ένα μόνο κελί, ή μία ορθογώνια περιοχή κελιών, οποιουδήποτε μεγέθους μέχρι και ολόκληρο φύλλο εργασίας. Η σύνταξη που χρησιμοποιούμε για να αναφερθούμε σε μία περιοχή κελιών, για παράδειγμα στην περιοχή A1:C5, είναι η παρακάτω:

Range("A1:C5") ή Range("A1", "C5")

Στην περίπτωση που αναφερόμαστε σε ένα μόνο κελί:

Range("A1")

Αν θέλουμε να αναφερθούμε σε μία ή περισσότερες στήλες, για παράδειγμα στις στήλες A έως C:

Range("A:C")

Αντίστοιχα για να αναφερθούμε σε μία ή περισσότερες γραμμές, για παράδειγμα στις γραμμές 4 έως 7:

Range("4:7")

Αν θέλουμε να αναφερθούμε σε μη συνεχόμενες περιοχές κελιών, χρησιμοποιούμε την παρακάτω σύνταξη:

Range("A1, C5, D8:E10")

Εδώ αναφερόμαστε στα κελιά A1, C5 και στην περιοχή D8:E10.

Μπορούμε επίσης να χρησιμοποιήσουμε ονόματα που έχουμε δώσει σε κελιά ή περιοχές κελιών. Αν για παράδειγμα έχουμε ονομάσει "Περιοχή1" την περιοχή B3:C5, τότε αναφερόμαστε σε αυτή ως εξής:

Range("Περιοχή1")

Στις επόμενες παραγράφους παρουσιάζονται οι βασικότερες ενέργειες που μπορούμε να πραγματοποιήσουμε σε περιοχές κελιών με το αντικείμενο **Range**.

Η ιδιότητα Value του αντικειμένου Range μας επιτρέπει να ορίσουμε ή να διαβάσουμε την τιμή των κελιών μίας περιοχής. Για να δώσουμε στο κελί A1 την τιμή 10 γράφουμε:

Range("A1").Value = 10

Το ίδιο ακριβώς αποτέλεσμα έχουμε αν γράψουμε:

Range("A1") = 10

Η παρακάτω γραμμή κώδικα έχει ως αποτέλεσμα να δοθεί η ίδια τιμή σε μία περιοχή (από A1 έως B3)που έχει παραπάνω από ένα κελιά:

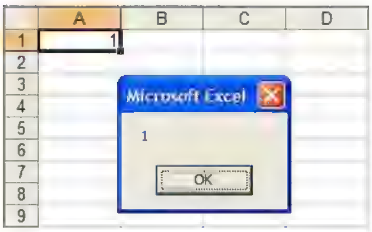
Range("A1:B3") = 10

Αντίθετα η γραμμή κώδικα να διαβάσουμε την τιμή ενός κελιού, για παράδειγμα:

MsgBox Range("A1").Value

μας επιτρέπει να διαβάσουμε το περιεχόμενο στου κελιού A1 με τη βοήθεια ενός Message Box.

Μετά την εκτέλεση του παραπάνω κώδικα εμφανίζεται το πλαίσιο διαλόγου, όπου φαίνεται η τιμή του κελιού A1.

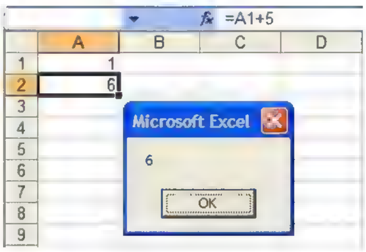


Εικόνα 37

Μπορούμε επίσης να χρησιμοποιήσουμε την ιδιότητα Value για να εισάγουμε ένα τύπο (formula) σε ένα ή περισσότερα κελιά. Ο παρακάτω κώδικας εισάγει στο κελί A2 τον τύπο "=A1+5" και στη συνέχεια εμφανίζει την τιμή του κελιού:

Range("A2").Value = "=A1+5"
MsgBox Range("A2").Value

Παρατηρούμε ότι στο κελί A2 υπάρχει ο τύπος "=A1+5", ενώ το πλαίσιο διαλόγου της **MsgBox** μας δείχνει την τιμή του κελιού που είναι το 6.

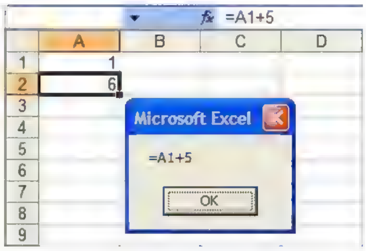


Εικόνα 38

Αν θέλουμε να δούμε τον τύπο χρησιμοποιούμε την ιδιότητα **Formula**:

Range("A2").Formula = "=A1+5"
MsgBox Range("A2").Value

Τώρα η **MsgBox** μας εμφανίζει το τύπο του κελιού A2:



Εικόνα 39

Με την ιδιότητα Formula μπορούμε να εισάγουμε και τύπους σε ένα ή περισσότερα κελιά.

Range("A2").Formula = "=A1+5"

Επιλογή, αντιγραφή, μετακίνηση και διαγραφή περιοχής κελιών

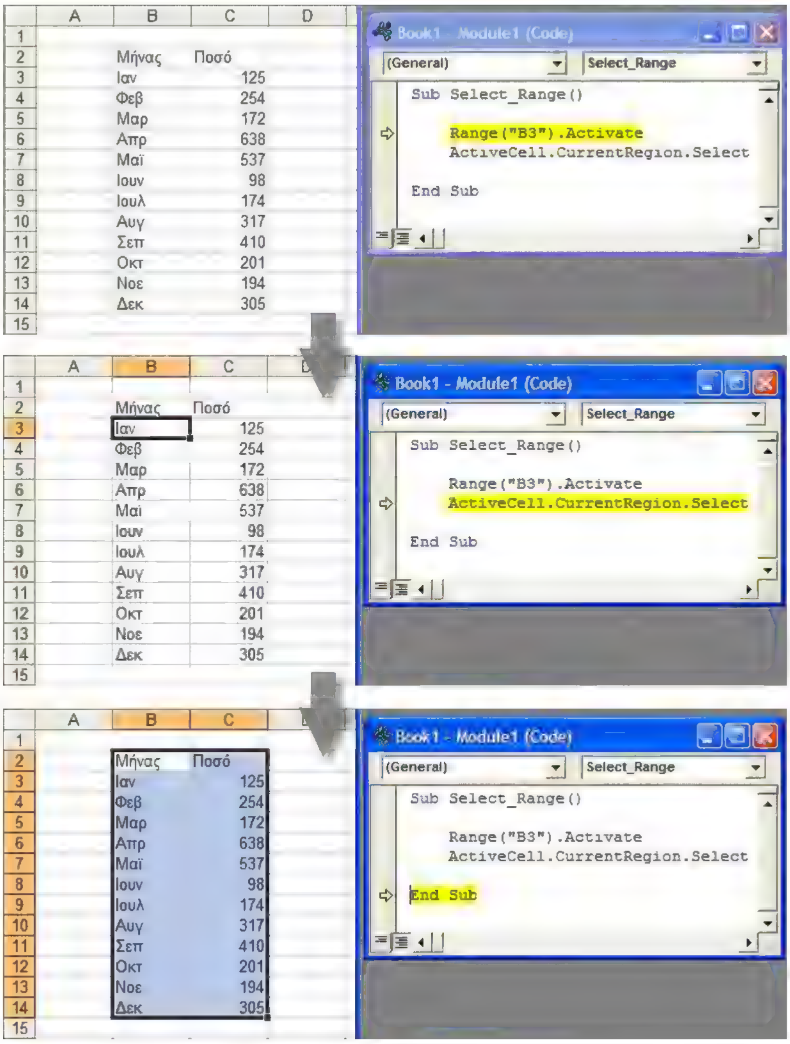
Για να επιλέξουμε μία περιοχή κελιών (για παράδειγμα την περιοχή A1:C5) χρησιμοποιούμε τη μέθοδο **Select**. Αυτό γίνεται ως εξής:

Range("A1:C5").Select

Αν σε ένα φύλλο έχουμε μία περιοχή συνεχόμενων κελιών που όμως δεν γνωρίζουμε τα όρια της, αλλά είμαστε βέβαιοι για κάποιο κελί που βρίσκεται μέσα στην περιοχή (για παράδειγμα το B3), μπορούμε να επιλέξουμε ως εξής:

Range("B3").Activate
ActiveCell.CurrentRegion.Select

Με την πρώτη εντολή ενεργοποιούμε το κελί B3, το οποίο πλέον είναι η τιμή του αντικειμένου **ActiveCell**. Στη συνέχεια με την ιδιότητα **CurrentRegion** βρίσκουμε την περιοχή συνεχόμενων κελιών που έχουν περιεχόμενο το εν λόγω κελί. Η ιδιότητα **CurrentRegion** επιστρέφει την περιοχή των συνεχόμενων κελιών, την οποία και επιλέγουμε με τη μέθοδο **Select**.

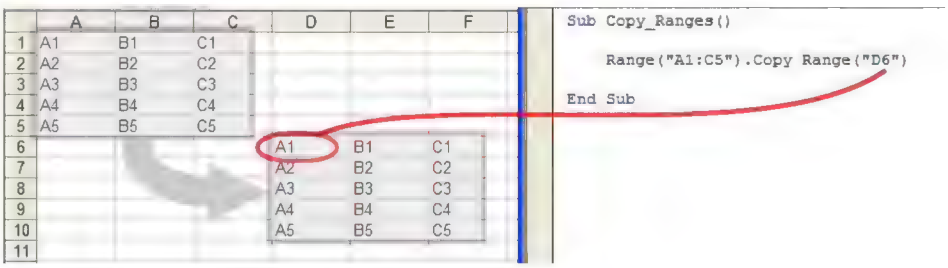


Εικόνα 40 Επιλογή περιοχής κελιών με την ιδιότητα CurrentRegion

Για να αντιγράψουμε μία περιοχή κελιών σε μία άλλη περιοχή (για παράδειγμα την περιοχή A1:C5 στην περιοχή D6:F10), χρησιμοποιούμε τη μέθοδο **Copy**:

Range("A1:C5").Copy Range("D6")

Παρατηρούμε ότι για την περιοχή στην οποία αντιγράφεται η αρχική περιοχή χρησιμοποιούμε μόνο το πάνω αριστερό κελί, στο παράδειγμά μας το D6.



Εικόνα 41 Αντιγραφή περιοχής κελιών

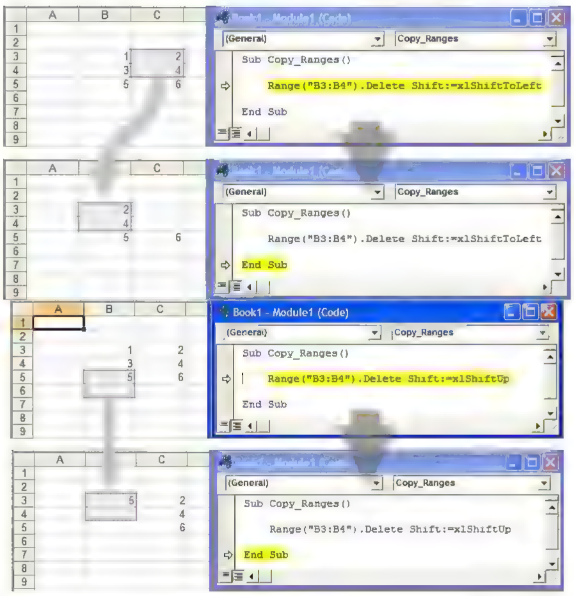
Με παρόμοιο τρόπο μετακινούμε μια περιοχή κελιών, με τη μέθοδο **Cut**:

Range("A1:C5").Cut Range("D6")

Προκειμένου να διαγράψουμε μία περιοχή κελιών χρησιμοποιούμε τη μέθοδο **Delete**:

Range("B3:B4").Delete Shift:=xlShiftUp

Η μέθοδος **Delete** διαγράφει την περιοχή των κελιών και όχι το περιεχόμενο τους. Με τη μέθοδο αυτή μπορούμε μέσω της προαιρετικής παραμέτρου **Shift** να ορίσουμε πως θα μετακινηθούν τα κελιά για να αντικαταστήσουν τα κελιά που διεγράφησαν. Η παράμετρος μπορεί να δεχτεί δύο τιμές: **xlShiftUp** ή **xlShiftToLeft**.

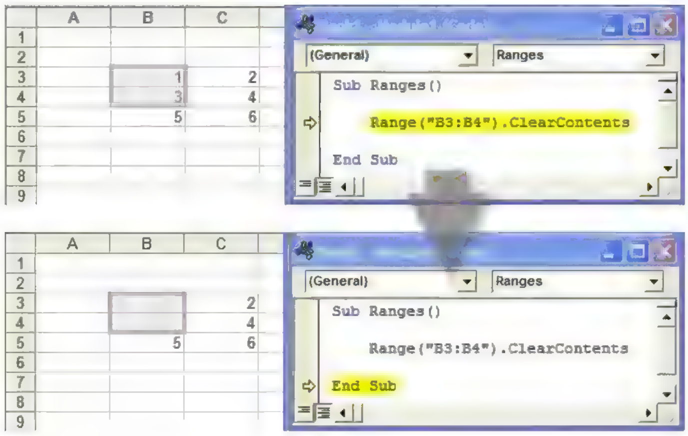


Εικόνα 42 Διαγραφή κελιών

Για να διαγράψουμε το περιεχόμενο περιοχής κελιών χρησιμοποιούμε τη μέθοδο **ClearContents**:

Range("B3:B4").ClearContents

Αντίστοιχα για να διαγράψουμε σχόλια και μορφοποίηση χρησιμοποιούμε τις μεθόδους **ClearComments** και **ClearFormats**, ενώ η μέθοδος **Clear** διαγράφει περιεχόμενο, μορφοποίηση και σχόλια.



Εικόνα 43 Διαγραφή περιεχομένου κελιών

Μορφοποίηση περιοχής κελιών

Η μορφοποίηση της γραμματοσειράς σε μία περιοχή κελιών ελέγχεται από το αντικείμενο **Font** του αντικειμένου **Range**. Το αντικείμενο Font έχει μία σειρά ιδιοτήτων, όπως **Bold**, **Italic**, **Size**, **Color** και άλλες, όπου οι τιμές τους αντιστοιχούν σε διάφορες μορφοποιήσεις. Για παράδειγμα:

Range("A1:A2").Font.Bold = True

τονίζει (κάνει bold) τη γραμματοσειρά (για να καταργηθεί ο τονισμός δίνουμε στην ιδιότητα **Bold** την τιμή False).

Range("A1:A2").Font.Size = 12

ρυθμίζει το μέγεθος της γραμματοσειράς σε 12.

Range("A1:A2").Font.Color = RGB(0, 255, 0)

ρυθμίζει το χρώμα σε πράσινο.

Range("A1:A2").Font.Name = "Arial Black"

ορίζει τη γραμματοσειρά σε Arial Black.

Τα αντικείμενα **Columns** και **Rows** που ανήκουν στο **Range** έχουν τις ιδιότητες **ColumnWidth** και **RowHeight** αντίστοιχα, μέσω των οποίων μπορούμε να ορίσουμε πλάτος στηλών και ύψος γραμμών:

Range("B3:C4").Columns.ColumnWidth = 20
Range("B3:C4").Rows.RowHeight = 20

Η μέθοδος **AutoFit** μας δίνει τη δυνατότητα να ορίσουμε το πλάτος ή το ύψος έτσι ώστε να προσαρμόζονται αυτόματα στο περιεχόμενο των κελιών της περιοχής. Για παράδειγμα:

Range("B3:C4").Columns.AutoFit

Μπορούμε επίσης να αναφερθούμε σε μία ή περισσότερες στήλες ή γραμμές, ως εξής:

```
Columns("A:C").Select
```

ή

```
Rows("5:10").EntireRow.AutoFit
```

Η ιδιότητα Cells

Αν θέλουμε να εργαστούμε με ένα κελί μπορούμε να χρησιμοποιήσουμε την ιδιότητα Cells, που ανήκει στα αντικείμενα **Application**, **ActiveSheet** και **Range**. Για παράδειγμα αν θέλουμε να επιλέξουμε το κελί B3 θα γράψουμε:

```
Cells(3,2)
```

Στη γενική μορφή γράφουμε Cells(x,y) όπου:

- ο πρώτος αριθμός x δηλώνει την γραμμή (1 έως 65536) και
- ο δεύτερος αριθμός y τη στήλη (1 έως 256).

Το πλεονέκτημα της χρήσης της Cells αντί του Range είναι ότι μπορούμε να χρησιμοποιήσουμε μεταβλητές, για να δίνουμε τις τιμές που θέλουμε στις παραμέτρους. Για παράδειγμα αν θέλουμε να δώσουμε στα κελιά A6 έως A10 τιμές σύμφωνα με την παρακάτω ακολουθία $3*N+N^2$, όπου N αριθμός που αντιστοιχεί στη γραμμή του κελιού (N=1 για το κελί A6, N=2 για το κελί A7 κλπ), μπορούμε να γράψουμε την παρακάτω υπορουτίνα:

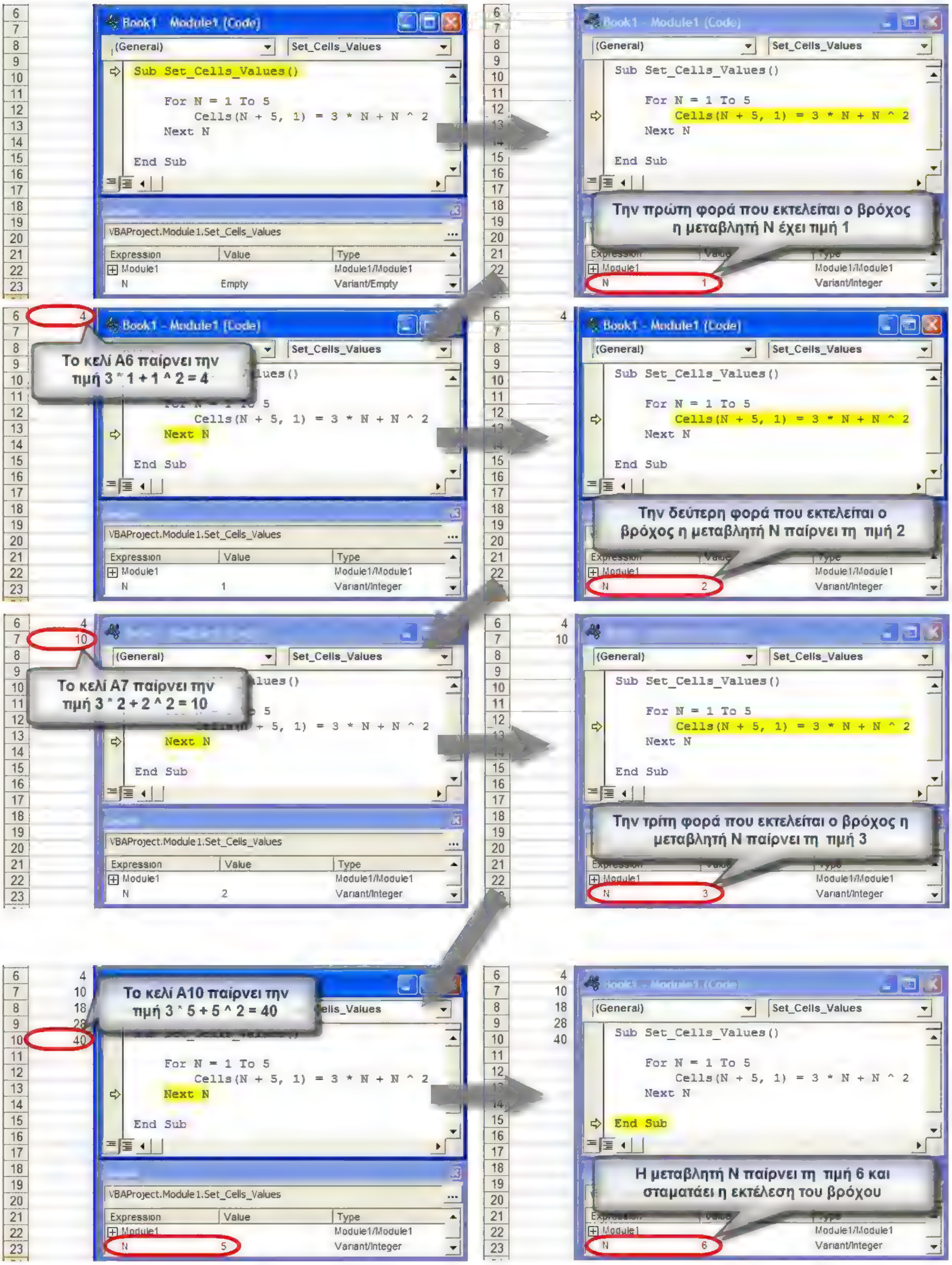
```
Sub Set_Cells_Values()

    For N = 1 To 5
        Cells(N + 5, 1) = 3 * N + N ^ 2
    Next N

End Sub
```

Η εντολή **For N = 1 To 5** συνδυάζεται με την εντολή **Next N** και δημιουργεί ένα **βρόχο** (**loop**) ο οποίος επαναλαμβάνεται (θα αναφερθούμε διεξοδικά στους βρόχους στη σελίδα 70). Οι εντολές που υπάρχουν ανάμεσα σε αυτές τις δύο, επαναλαμβάνονται όσες φορές ορίζεται στην For (στο παράδειγμα 5 φορές). Σε κάθε επανάληψη η τιμή του N αυξάνεται κατά 1. Την πρώτη φορά που εκτελείται η εντολή **Cells(N + 5, 1) = 3 * N + N ^ 2**, το N έχει την τιμή 1, επομένως το κελί θα είναι το Cells(1+5 ,1) δηλαδή το A6, και η τιμή του 3 * 1 + 1 ^ 2. Σε κάθε διαδοχική εκτέλεση μεταβάλλεται η τιμή της μεταβλητής N και επομένως αλλάζει τόσο το κελί, όσο και η τιμή που θα γράφεται σε αυτό. Αν εκτελούμε βηματικά τον κώδικα μπορούμε στο Locals Window να βλέπουμε πως μεταβάλλεται η τιμή της μεταβλητής N.

Όταν όμως γράψουμε **Application.Cells** ή **ActiveSheet.Cells** η ιδιότητα **Cells** επιστρέφει ένα αντικείμενο **Range** που περιέχει όλα τα κελιά του ενεργού φύλλου.



Εικόνα 44 Η ιδιότητα Cells σε βρόχο

Περιοχές κελιών και πίνακες

Μπορούμε να αποθηκεύσουμε τα περιεχόμενα μίας περιοχής κελιών σε μία μεταβλητή τύπου Variant, όπως στο επόμενο παράδειγμα:

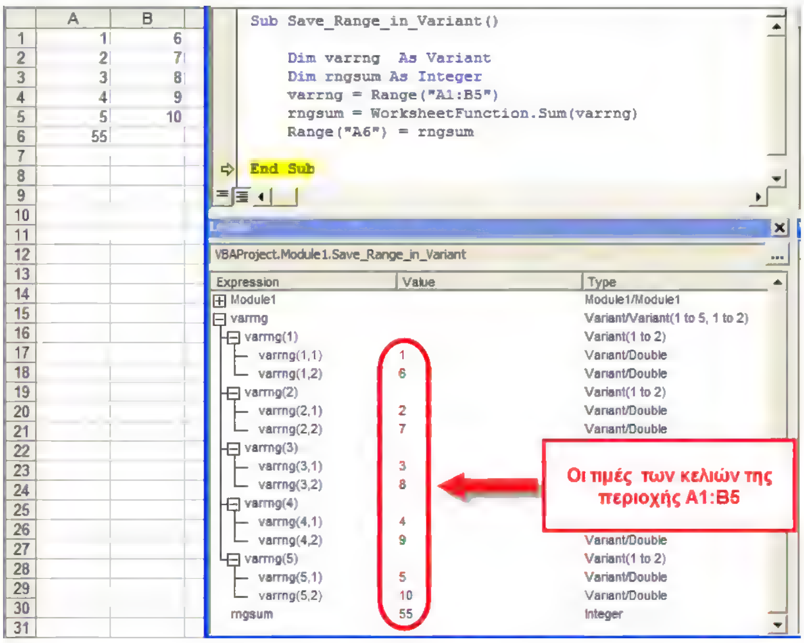
```
Sub Save_Range_in_Variant ()

  Dim varrng As Variant
  Dim rngsum As Integer

  varrng = Range("A1:B5")
  rngsum = WorksheetFunction.Sum(varrng)
  Range("A6") = rngsum

End Sub
```

Εδώ αποθηκεύουμε με την έκφραση **varrng = Range("A1:B5")**, τις τιμές των κελιών της περιοχής A1:A5 στη μεταβλητή **varrng** και στη συνέχεια στο κελί A6 το άθροισμα των τιμών.



Εικόνα 45 Η αποθήκευση των τιμών της περιοχής κελιών στη μεταβλητή τύπου Variant

Εισαγωγή στο Excel Object Model

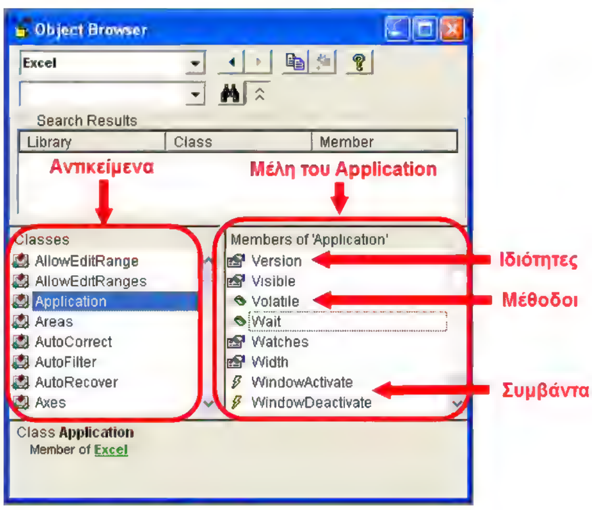
Η VBA χρησιμοποιεί αντικείμενα για να ελέγχει το Excel. Είτε δουλεύουμε με την εφαρμογή του Excel (αντικείμενο **Application**), είτε με κάποιο βιβλίο εργασίας (αντικείμενο **Workbook**), είτε με μία περιοχή κελιών (αντικείμενο **Range**), χειριζόμαστε κάποιο αντικείμενο. Το Excel έχει πάνω από 200 αντικείμενα τα οποία παρουσιάζουν περί τις 5000 ιδιότητες και μεθόδους. Κάθε αντικείμενο αντιπροσωπεύει ένα στοιχείο του Excel. Κάθε αντικείμενο είτε περιέχει άλλα αντικείμενα, είτε περιέχεται σε άλλα αντικείμενα. Δηλαδή τα αντικείμενα το Excel τοποθετούνται σε μία ιεραρχική δομή που είναι το **Excel Object Model**. Το Excel Object Model περιγράφει τις σχέσεις μεταξύ των αντικειμένων του Excel.

Τα αντικείμενα που χρησιμοποιούνται κατά κύριο λόγο είναι τα:

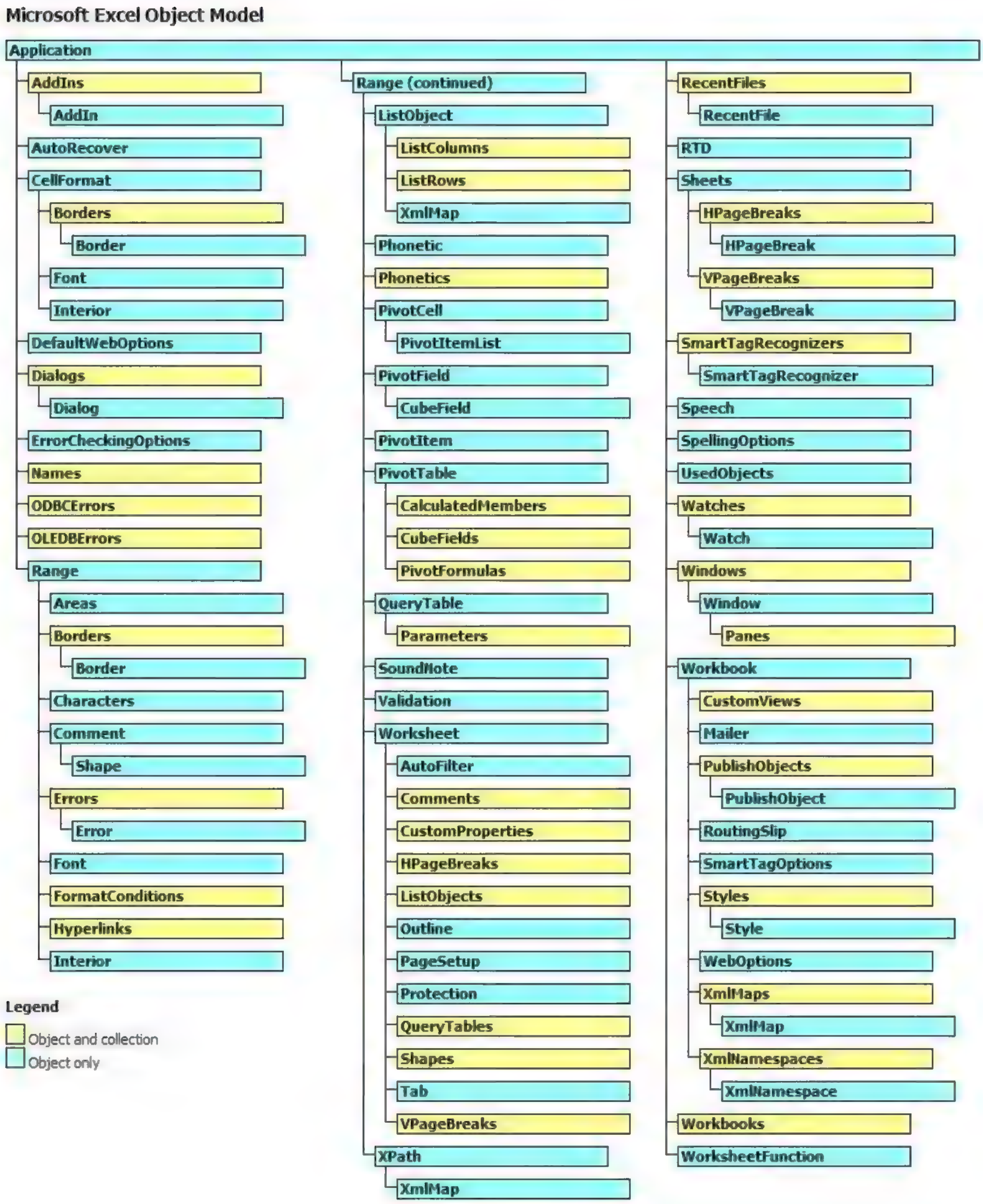
- Application
- Workbook
- Worksheet
- Range
- Chart
- Dialog

Παρόμοια αντικείμενα εντάσσονται σε συλλογές. Για παράδειγμα η συλλογή **Worksheets** απαρτίζεται από τα αντικείμενα **Worksheet**, που είναι τα φύλλα ενός βιβλίου εργασίας. Κάθε αντικείμενο που περιέχει περισσότερα από ένα όμοια αντικείμενα είναι ταυτόχρονα και συλλογή. Για κάθε αντικείμενο υπάρχει και μία κλάση.

Ο επεξεργαστής της Visual Basic έχει ένα ιδιαίτερα χρήσιμο εργαλείο, τον Object Browser, που παρέχει πληροφορίες για τα αντικείμενα, τι ιδιότητες, τις μεθόδους και τα συμβάντα. Ανοίγουμε τον Object Browser, από το **View - Object Browser**.



Εικόνα 46 Ο Object Browser



Εικόνα 47 Το Excel Object Model

Το αντικείμενο Application

Το πρώτο στην ιεραρχία αντικείμενο είναι το **Application**, που για το Excel είναι το ίδιο το Excel. Περιέχει όλες τις μεθόδους και ιδιότητες που είναι απαραίτητες για να προγραμματίσουμε μέσω της VBA το Excel, καθώς και όλα τα υπόλοιπα αντικείμενα. Οι πιο χρήσιμες ιδιότητές του είναι:

Ιδιότητα	
ActiveCell	Επιστρέφει ένα αντικείμενο **Range** που αναφέρεται στο ενεργό κελί του ενεργού φύλλου εργασίας
ActiveChart	Επιστρέφει ένα αντικείμενο **Chart** που αναφέρεται στο ενεργό γράφημα του ενεργού φύλλου εργασίας
ActiveSheet	Επιστρέφει ένα αντικείμενο **Worksheet** που αναφέρεται στο ενεργό φύλλο εργασίας
ActiveWindow	Επιστρέφει το ενεργό παράθυρο στην εφαρμογή του Excel
ActiveWorkbook	Επιστρέφει ένα αντικείμενο **Workbook** που αναφέρεται στο ενεργό βιβλίο εργασίας
RangeSelection	Επιστρέφει ένα αντικείμενο **Range** που αναφέρεται στην επιλεγμένη περιοχή κελιών
Selection	Επιστρέφει ένα αντικείμενο **Chart**, ή ένα αντικείμενο **Range** (ανάλογα με το τι έχουμε επιλέξει στο ενεργό φύλλο εργασίας)
ScreenUpdating	Όταν εκτελούνται εντολές σε κώδικα VBA, το Excel προσπαθεί να ανταποκριθεί σε αυτές ανανεώνοντας την οθόνη. Αν δώσουμε στην ιδιότητα αυτή τη τιμή False, η οθόνη του Excel δεν αντικατοπτρίζει τυχόν αλλαγές που επιφέρει η εκτέλεση των εντολών.
StatusBar	Επιτρέπει την εμφάνιση μηνυμάτων στη γραμμή κατάστασης
ThisWorkbook	Αναφέρεται στο βιβλίο εργασίας που περιέχει τον κώδικα που εκτελείται

Πίνακας 3 Χρησιμότερες ιδιότητες του αντικειμένου Application

Στο παρακάτω παράδειγμα εμφανίζουμε στη γραμμή κατάστασης ένα μήνυμα:

Application.StatusBar = "ΕΘΝΙΚΟ ΚΕΝΤΡΟ ΔΗΜΟΣΙΑΣ ΔΙΟΙΚΗΣΗΣ"



Εικόνα 48 Εμφάνιση μηνύματος στη γραμμή κατάστασης

Το μήνυμα αυτό θα εμφανίζεται, ακόμη και όταν εμείς εκτελούμε ενέργειες, που κανονικά θα επηρέαζαν τη γραμμή κατάστασης, η οποία θα επανέλθει μόνο όταν εκτελέσουμε την παρακάτω εντολή:

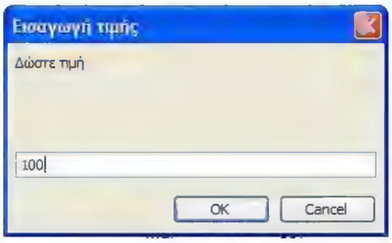
Application.StatusBar = False

Δύο χρήσιμες μέθοδοι είναι η **FindFile** και η **InputBox**. Με την πρώτη δίνουμε εντολή στο Excel να εμφανίσει το παράθυρο διαλόγου **Άνοιγμα**, προκειμένου να ανοίξουμε ένα αρχείο του Excel. Η δεύτερη εμφανίζει ένα παράθυρο διαλόγου, στο οποίο ο χρήστης μπορεί να εισάγει πληροφορία, που θα χρησιμοποιηθεί στον κώδικα. Στο επόμενο παράδειγμα εμφανίζουμε ένα παράθυρο διαλόγου όπου ο χρήστης εισάγει μία τιμή και αυτή αποδίδεται στη μεταβλητή **Alfa**:

Sub InputBox_Method()

Alfa = Application.InputBox(Prompt:="Δώστε τιμή", Title:="Εισαγωγή τιμής", Type:=1)

End Sub



Εικόνα 49 Εισαγωγή πληροφορίας με τη μέθοδο InputBox

Η συλλογή Workbooks

Η συλλογή **Workbooks,** αποτελείται από αντικείμενα **Workbook** και περιέχει αναφορά σε κάθε ανοικτό βιβλίο εργασίας του Excel. Μέσω των μεθόδων που έχει μπορούμε να δημιουργήσουμε ένα νέο βιβλίο εργασίας, να κλείσουμε ένα ανοικτό βιβλίο ή να ανοίξουμε ένα βιβλίο που έχουμε αποθηκευμένο.

Για να δημιουργήσουμε ένα βιβλίο εργασίας χρησιμοποιούμε τη μέθοδο **Add**, ως εξής:

```
Workbooks.Add
```

Στο επόμενο παράδειγμα ανοίγουμε το βιβλίο test.xls που είναι αποθηκευμένο στο φάκελο My Documents, χρησιμοποιώντας τη μέθοδο **Open**.

```
Sub Open_Workbook()

   Dim wrkbk As Workbook
   Set wrkbk = Workbooks.Open(Filename:="test.xls")

End Sub
```

Εδώ χρησιμοποιούμε τη μεταβλητή **wrkbk** που είναι τύπου αντικειμένου **Workbook** (object variable). Οι μεταβλητές αντικειμένων δηλώνονται όπως και οι συνηθισμένες μεταβλητές, διαφέρει όμως ο τρόπος που αποδίδουμε σε αυτές τιμές. Ενώ για τις τελευταίες χρησιμοποιούμε μόνο το σύμβολο ισότητας, για τις μεταβλητές αντικειμένων χρησιμοποιούμε ισότητα σε συνδυασμό και με την έκφραση **Set**:

```
Set variablename = variablevalue
```

όπου **variablename** το όνομα της μεταβλητής και **variablevalue** η τιμή που της αποδίδουμε.

Η μέθοδος **Open** έχει πολλές παραμέτρους, αλλά εδώ χρησιμοποιούμε μόνο την Filename, στην οποία πρέπει να δώσουμε το όνομα του βιβλίου που θέλουμε να ανοίξουμε.

Η χρήση της μεθόδου **Close** έχει σαν αποτέλεσμα το κλείσιμό ένος βιβλόυ . Για παράδειγμα, προκειμένου να κλείσουμε το βιβλίο test.xls που είχαμε ανοίξει προηγουμένως γράφουμε:

```
Sub Close_Workbook()
    Workbooks("test.xls").Close
End Sub
```

Στην περίπτωση όπου υπάρχουν αλλαγές, το Excel εμφανίζει μήνυμα και μας ρωτάει αν θέλουμε να τις αποθηκεύσουμε. Μπορούμε όμως να ορίσουμε με την παράμετρο **SaveChanges** αν θέλουμε να αποθηκευτούν οι αλλαγές:

```
Sub Close_Workbook()
    Workbooks("test.xls").Close SaveChanges:=True
End Sub
```

Αν δώσουμε στην παράμετρο **SaveChanges** την τιμή **False**, το Excel αποθηκεύει το βιβλίο χωρίς τις αλλαγές.

Το αντικείμενο Workbook

Το αντικείμενο **Workbook** αναφέρεται σε ένα βιβλίο εργασίας και έχει περισσότερες ιδιότητες και μεθόδους από τη συλλογή **Workbooks**. Οι πιο χρήσιμες ιδιότητες είναι:

Ιδιότητα	
ActiveChart	Επιστρέφει ένα αντικείμενο **Chart** που αναφέρεται στο ενεργό γράφημα
ActiveSheet	Επιστρέφει ένα αντικείμενο **Worksheet** που αναφέρεται στο ενεργό φύλλο εργασίας
FullName	Επιστρέφει το όνομα και τη διαδρομή που είναι αποθηκευμένο το βιβλίο
Name	Επιστρέφει το όνομα του βιβλίου
Password	Αποδίδει ή επιστρέφει το password που θέλουμε να δώσουμε ή ήδη έχει ένα βιβλίο
Path	Επιστρέφει τη διαδρομή που είναι αποθηκευμένο το βιβλίο
Sheets	Επιστρέφει μία συλλογή αντικειμένων **Worksheets** αποτελούμενη από τα φύλλα που περιέχει το βιβλίο
Worksheets	Επιστρέφει μία συλλογή αντικειμένων **Worksheets** αποτελούμενη από τα φύλλα που περιέχει το βιβλίο

Πίνακας 4 Χρησιμότερες ιδιότητες του αντικειμένου Workbook

Οι πιο χρήσιμες μέθοδοι του αντικειμένου Workbook είναι οι παρακάτω:

Μέθοδος	
Activate	Ενεργοποιεί το βιβλίο
Close	Κλείνει το βιβλίο
PrintOut	Εκτυπώνει το βιβλίο
PrintPreview	Δημιουργεί προεπισκόπηση εκτύπωσης του βιβλίου
Protect	Ορίζει αν το βιβλίο θα προστατεύεται από αλλαγές ή όχι
Save	Αποθηκεύει το βιβλίο με το ίδιο όνομα
SaveAs	Αποθηκεύει το βιβλίο με νέο όνομα
SaveCopyAs	Αποθηκεύει ένα αντίγραφο του βιβλίου

Πίνακας 5 Χρησιμότερες μέθοδοι του αντικειμένου Workbook

Για να αναφερθούμε στο βιβλίο εργασίας, στο οποίο γράφουμε κώδικα VBA, χρησιμοποιούμε την ιδιότητα **ThisWorkbook** του αντικειμένου **Application**. Η ιδιότητα αυτή επιστρέφει ένα αντικείμενο **Workbook** που αντιστοιχεί στο βιβλίο που περιέχει τον κώδικα που εκτελούμε. Αν για παράδειγμα έχουμε ανοικτά τα βιβλία Βιβλίο1.xls και Βιβλίο2.xls, και σε μία υπορουτίνα που βρίσκεται στο Βιβλίο1.xls γράψουμε:

```
MsgBox ThisWorkbook.Name
```

όταν την εκτελέσουμε θα εμφανιστεί ένα πλαίσιο διαλόγου με ένα μήνυμα που θα γράφει "Βιβλίο1".

Αν θέλουμε να τυπώσουμε όλα τα φύλλα (που έχουν περιεχόμενο) ενός βιβλίου εργασίας, μπορούμε να γράψουμε τη παρακάτω υπορουτίνα:

```
Sub Print_Workbook

   ThisWorkbook.PrintOut From:=1, To:=2, Copies:=2, _
 ActivePrinter:="LexmarkT642"

End Sub
```

Εκτυπώνουμε τις δύο πρώτες σελίδες (**From:=1, To:=2**) σε δύο αντίγραφα (**Copies:=2**) στον εκτυπωτή **LexmarkT642**. Αν δεν δώσουμε τιμή σε καμία παράμετρο, θα εκτυπωθούν όλες οι σελίδες στον προκαθορισμένο από το λειτουργικό σύστημα εκτυπωτή.

Η επόμενη υπορουτίνα αποθηκεύει το βιβλίο εργασίας με νέο όνομα στη διαδρομή "C:\VBA Files":

```
Sub Save_Workbook

   ThisWorkbook.SaveAs FileName:="C:\VBA Files\Saved_VBA_Example.xls"

End Sub
```

Το αντικείμενο Workbook έχει μερικά ιδιαίτερα χρήσιμα **συμβάντα** (**events**) με τα οποία μπορούμε να εκτελέσουμε σημαντικές εργασίες, που αλλιώς θα ήταν αδύνατον να εκτελέσουμε.

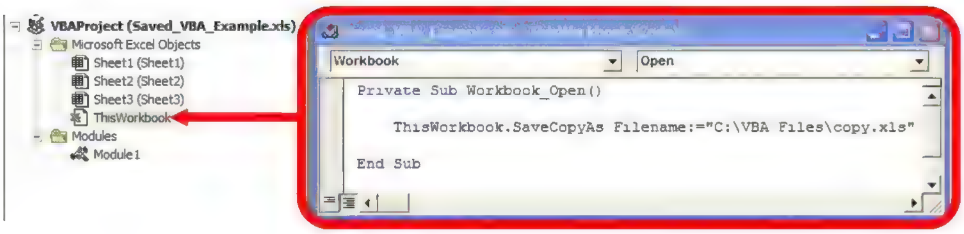
Συμβάν	
Activate	Εκτελεί τον κώδικα όταν ενεργοποιείται το βιβλίο
BeforeClose	Εκτελεί τον κώδικα πριν κλείσει το βιβλίο
BeforePrint	Εκτελεί τον κώδικα πριν εκτυπωθεί το βιβλίο
BeforeSave	Εκτελεί τον κώδικα πριν αποθηκευτεί το βιβλίο
Deactivate	Εκτελεί τον κώδικα όταν απενεργοποιείται το βιβλίο
NewSheet	Εκτελεί τον κώδικα όταν εισαχθεί νέο φύλλο εργασίας
Open	Εκτελεί τον κώδικα μόλις ανοίξει το βιβλίο
SheetActivate	Εκτελεί τον κώδικα μόλις ενεργοποιηθεί οποιοδήποτε φύλλο του βιβλίου
SheetChange	Εκτελεί τον κώδικα μόλις γίνει κάποια αλλαγή σε οποιοδήποτε φύλλο του βιβλίου
SheetDeactivate	Εκτελεί τον κώδικα μόλις απενεργοποιηθεί το ενεργό φύλλο του βιβλίου

Πίνακας 6 Χρήσιμα συμβάντα του αντικειμένου Workbook

Τα συμβάντα μας επιτρέπουν να εκτελούμε εντολές, παρακολουθώντας τι συμβαίνει στην εφαρμογή μας. Για παράδειγμα αν θέλουμε να αποθηκεύουμε ένα αντίγραφο ασφαλείας με το όνομα copy.xls στη διαδρομή "C:\VBA Files", κάθε φορά που ανοίγουμε το βιβλίο εργασίας, γράφουμε την επόμενη υπορουτίνα:

```
Private Sub Workbook_Open()
    ThisWorkbook.SaveCopyAs "C:\VBA Files\copy.xls"
End Sub
```

Βλέπουμε ότι η υπορουτίνα έχει το όνομα Workbook_Open, δηλαδή συνδυάζει το αντικείμενο (**Workbook**) και το συμβάν (**Open**). Τα δύο αυτά ενώνονται με το χαρακτήρα "_". Επίσης η υπορουτίνα δεν βρίσκεται σε κάποιο module, αλλά στο αντικείμενο ThisWorkbook του βιβλίου εργασίας, όπως φαίνεται και στον Project Explorer:



Εικόνα 50 Ο κώδικας για το συμβάν Open του αντικειμένου Workbook

Οι συλλογές Worksheets και Sheets

Η συλλογή **Worksheets** αποτελείται από αντικείμενα **Worksheet** και περιέχει αναφορά σε κάθε φύλλο εργασίας ενός βιβλίου. Η συλλογή **Sheets** αποτελείται από αντικείμενα **Worksheet** και **Chart** και περιέχει αναφορά σε κάθε φύλλο εργασίας καθώς και στα φύλλα γραφημάτων ενός βιβλίου.

Οι σημαντικότερες ιδιότητες τους είναι η **Count** και η **Visible**. Η πρώτη επιστρέφει τον αριθμό των φύλλων που έχει ένα βιβλίο (για τη συλλογή **Worksheets** τον αριθμό μόνο των φύλλων εργασίας, ενώ για τη **Sheets** των φύλλων εργασίας και των φύλλων γραφήματος) και η δεύτερη ορίζει αν τα φύλλα είναι ορατά ή όχι. Στο επόμενο παράδειγμα βρίσκουμε τον αριθμό των φύλλων εργασίας που περιέχει το βιβλίο (η μεταβλητή **sheet_count**), και στη συνέχεια αποκρύπτουμε το τελευταίο. Για να αναφερθούμε στο τελευταίο φύλλο εργασίας χρησιμοποιούμε ως δείκτη στη **Worksheets**, τη μεταβλητή **sheet_count**, δηλαδή τον αριθμό των φύλλων που βρήκαμε ότι έχει το βιβλίο.

```
Sub hide_last_sheet()

    Dim sheet_count As Integer
    sheet_count = ThisWorkbook.Worksheets.Count
    ThisWorkbook.Worksheets(sheet_count).Visible = False

End Sub
```

Οι χρησιμότερες μέθοδοι των συλλογών **Worksheets** και **Sheets** φαίνονται στον παρακάτω πίνακα:

Μέθοδος	
Add	Προσθέτει ένα νέο φύλλο στη συλλογή
Copy	Αντιγράφει ένα φύλλο της συλλογής
Delete	Διαγράφει ένα φύλλο της συλλογής
Move	Μετακινεί ένα φύλλο της συλλογής
PrintOut	Εκτυπώνει ένα φύλλο της συλλογής
PrintPreview	Δημιουργεί προεπισκόπηση εκτύπωσης του φύλλου της συλλογής
Select	Επιλέγει ένα φύλλο από τη συλλογή

Πίνακας 7 Χρησιμότερες μέθοδοι των συλλογών Worksheets και Sheets

Στο επόμενο παράδειγμα προσθέτουμε δύο νέα φύλλα εργασίας (**Count:=2**) πριν το τελευταίο φύλλο (**Before:=Worksheets(Worksheets.Count)**):

```
Sub Add_sheets()

    ActiveWorkbook.Worksheets.Add Before:=Worksheets(Worksheets.Count), _
    Count:=2

End Sub
```

Στο προηγούμενο παράδειγμα χρησιμοποιήσαμε μία μεταβλητή για να αποθηκεύσουμε τον αριθμό των φύλλων εργασίας του βιβλίου, και στη συνέχεια τη χρησιμοποιήσαμε ως δείκτη για να αναφερθούμε στο τελευταίο φύλλο του βιβλίου. Εδώ όμως χρησιμοποιήσαμε απευθείας την αντίστοιχη ιδιότητα του

αντικειμένου για να αναφερθούμε στο τελευταίο φύλλο: **Worksheets(Worksheets.Count)**. Αν θέλαμε να προσθέσουμε τα φύλλα μετά από κάποιο φύλλο, τότε αντί για την παράμετρο **Before** θα χρησιμοποιούσαμε την **After**.

Επίσης εδώ χρησιμοποιούμε την ιδιότητα **ActiveWorkbook** του αντικειμένου **Application** για να αναφερθούμε στο ενεργό βιβλίο εργασίας, που μπορεί να είναι το ίδιο ή διαφορετικό από αυτό στο οποίο βρίσκεται ο κώδικας που εκτελούμε.

Αν θέλουμε να αντιγράψουμε ένα φύλλο, χρησιμοποιούμε τη μέθοδο **Copy**:

```
Sub Copy_sheets()
   ActiveWorkbook.Worksheets(1).Copy After:=Worksheets(Worksheets.Count)
End Sub
```

Και εδώ μπορούμε να χρησιμοποιήσουμε την παράμετρο **Before** αντί της **After**. Αντίστοιχα λειτουργεί και η μέθοδος **Move**, η οποία μετακινεί ένα φύλλο, ενώ η μέθοδος **Delete** διαγράφει ένα φύλλο:

```
Sub Copy_sheets()
   ActiveWorkbook.Worksheets(1).Delete
End Sub
```

Οι μέθοδοι **PrintOut** και **PritPreview** λειτουργούν όπως και οι αντίστοιχες του αντικειμένου **Workbook**, αλλά αναφέρονται πλέον σε φύλλο και όχι σε ολόκληρο το βιβλίο εργασίας. Στο επόμενο παράδειγμα εκτυπώνουμε το πρώτο φύλλο εργασίας του βιβλίου.

```
Sub Print_sheet()
   ActiveWorkbook.Worksheets(1).PrintOut
End Sub
```

Αν το βιβλίο μας είχε και ένα φύλλο γραφήματος πριν το πρώτο φύλλο εργασίας, το προηγούμενο παράδειγμα θα τύπωνε το πρώτο φύλλο εργασίας και όχι το φύλλο γραφήματος, γιατί η συλλογή Worksheets δεν περιλαμβάνει φύλλα γραφήματος. Αν όμως γράφαμε **ActiveWorkbook.Sheets(1).PrintOut** τότε θα εκτυπώναμε το φύλλο γραφήματος.

Το αντικείμενο Worksheet

Το αντικείμενο **Worksheet** αναφέρεται σε ένα φύλλο εργασίας. Οι πιο χρήσιμες ιδιότητες του αντικειμένου είναι οι παρακάτω:

Ιδιότητα	
Index	Επιστρέφει τον δείκτη που αντιστοιχεί στο φύλλο εργασίας
Name	Ορίζει ή επιστρέφει το όνομα του φύλλο εργασίας
Names	Επιστρέφει μία συλλογή από αντικείμενα Range στα οποία έχουν αποδοθεί ονόματα
Next	Επιστρέφει το επόμενο φύλλο ως αντικείμενο Worksheet
Previous	Επιστρέφει το προηγούμενο φύλλο ως αντικείμενο Worksheet
Visible	Ορίζει αν το φύλλο είναι ορατό

Πίνακας 8 Χρησιμότερες ιδιότητες του αντικειμένου Worksheet

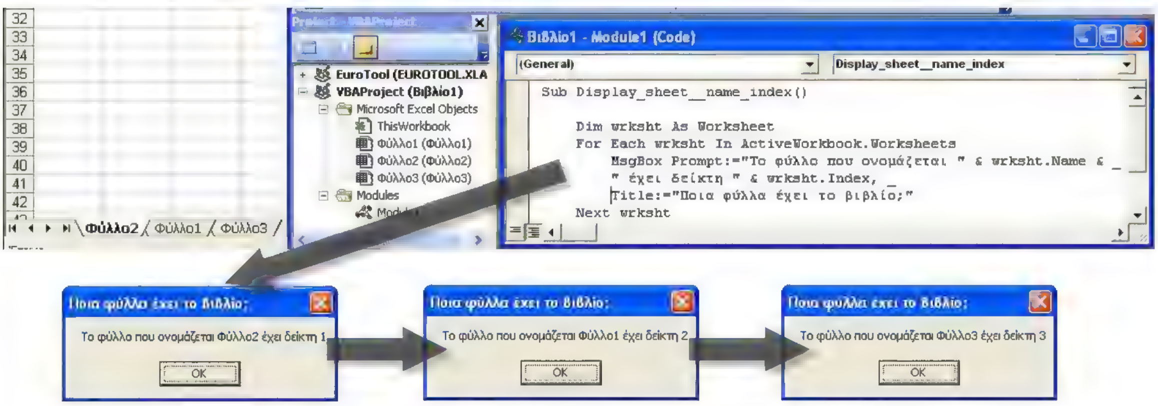
Στο παρακάτω παράδειγμα ανατρέχουμε σε όλα τα φύλλα εργασίας του ενεργού βιβλίου και εμφανίζουμε το όνομα και το δείκτη που αντιστοιχεί σε αυτά:

```
Sub Display_sheet_name_index()

   Dim wrksht As Worksheet
   For Each wrksht In ActiveWorkbook.Worksheets
      MsgBox Prompt:="Το φύλλο που ονομάζεται " & wrksht.Name & _
      " έχει δείκτη " &  wrksht.Index, _
      Title:="Ποια φύλλα έχει το βιβλίο;"
   Next wrksht

End Sub
```

Όπως φαίνεται στον παραπάνω κώδικα, έχουμε δηλώσει τη μεταβλητή **wrksht** ως μεταβλητή αντικειμένου **Worksheet**, και τη χρησιμοποιούμε για να εμφανίσουμε μέ το **MsgBox** μήνυμα με το όνομα και το δείκτη κάθε φύλλου εργασίας. Αυτό θα συμβεί για κάθε φύλλο του βιβλίου, λόγω του βρόγχου **For Each wrksht In ActiveWorkbook.Worksheets ... Next wrksht**, ο οποίος εκτελείται παρόμοια με τον απλό βρόχο **For N = X To Y... Next**. Η εντολή αυτή χρησιμοποιείται όταν έχουμε μία συλλογή αντικειμένων (στο παράδειγμά μας τη συλλογή **Worksheets**) και θέλουμε να εκτελέσουμε ένα σύνολο εντολών για κάθε αντικείμενο που ανήκει στη συλλογή (στο παράδειγμά μας για τα 3 φύλλα εργασίας που έχει η συλλογή **Worksheets** του ενεργού βιβλίου).



Εικόνα 51 Χρήση των ιδιοτήτων Name και Index του αντικειμένου Worksheet

Οι σημαντικότερες μέθοδοι του αντικειμένου **Worksheet** παρουσιάζονται στον επόμενο πίνακα:

Μέθοδος	
Activate	Ενεργοποιεί το φύλλο εργασίας
Copy	Αντιγράφει το φύλλο εργασίας
Delete	Διαγράφει το φύλλο εργασίας
Move	Μετακινεί το φύλλο εργασίας
Paste	Επικολλάει τα περιεχόμενα του Πρόχειρου (Clipboard) στο φύλλο εργασίας
PasteSpecial	Επικολλάει τα περιεχόμενα του Πρόχειρου (Clipboard) στο φύλλο εργασίας
PivotTable	Επιστρέφει μία **Αναφορά Συγκεντρωτικού Πίνακα** (**Pivot Table**) ή μία συλλογή από Αναφορές
PivotTableWizard	Δημιουργεί μία Αναφορά Συγκεντρωτικού Πίνακα
PrintOut	Εκτυπώνει το φύλλο εργασίας
PrintPreview	Δημιουργεί προεπισκόπηση εκτύπωσης του φύλλου
Protect	Προστατεύει το φύλλο εργασίας από αλλαγές
ResetAllPageBreaks	Απαλείφει τυχόν ορισμένες περιοχές εκτύπωσης
SaveAs	Αποθηκεύει το φύλλο εργασίας σε ένα νέο βιβλίο
Select	Επιλέγει το φύλλο εργασίας
Unprotect	Ακυρώνει τυχόν προστασία του φύλλο εργασίας από αλλαγές

Πίνακας 9 Χρησιμότερες μέθοδοι του αντικειμένου Worksheet

Οι μέθοδοι **Copy**, **Delete**, **Move**, **PrintOut**, **PrintPreview** λειτουργούν όπως οι αντίστοιχες μέθοδοι της συλλογής **Worksheets**. Η μέθοδος **Paste** κάνει επικόλληση των περιεχομένων του Πρόχειρου στη περιοχή κελιών (**Range**) που δηλώνουμε στην παράμετρο **Destination**.

Παράδειγμα: Έστω ότι έχουμε ένα βιβλίο εργασίας, όπου στα φύλλα του έχουν καταγραφεί οι ώρες υπερωριακής απασχόλησης υπαλλήλων και τα ονοματεπώνυμα τους. Θέλουμε να δημιουργήσουμε αρχείο μορφής html που θα περιέχει τα στοιχεία αυτά σε ένα πίνακα για όλους τους υπαλλήλους. Κάθε φύλλο περιέχει στοιχεία για ένα υπάλληλο και δεν γνωρίζουμε το συνολικό αριθμό τους. Ο κώδικας θα πρέπει να εκτελείται σωστά για οποιοδήποτε αριθμό υπαλλήλων, και κατά συνέπεια φύλλων του βιβλίου.

- Με την ιδιότητα **Count** της συλλογής **Worksheets** βρίσκουμε τον αριθμό των φύλλων που έχει το βιβλίο, και προσθέτουμε ένα νέο φύλλο, χρησιμοποιώντας τη μέθοδο **Add** της συλλογής.
- Στη συνέχεια με τις μεθόδους **Copy** του αντικειμένου **Range** και **Paste** του αντικειμένου **Worksheet**, αντιγράφουμε και επικολλούμε την περιοχή κελιών "A4:A16" από το πρώτο φύλλο στο νέο που δημιουργήσαμε.
- Μετά, για κάθε φύλλο του βιβλίου (όπως προηγουμένως με το βρόχο **For Each wrksht In ThisWorkbook.Worksheets ... Next wrksht**), εκτός από το νέο φύλλο (**If wrksht_id < ThisWorkbook.Worksheets.Count Then ... End If**), αντιγράφουμε και επικολλούμε τις περιοχές κελιών που περιέχουν τα στοιχεία που επιθυμούμε. Χρησιμοποιούμε τη μεταβλητή **wrksht_id**, η τιμή της οποίας ισούται με το δείκτη του φύλλου εργασίας για το οποίο εκτελείται ο βρόχος. Η τιμή της αυξάνει κατά ένα, κάθε φορά που εκτελείται ο βρόχος.

- Μετά την έξοδο από το πρώτο βρόχο, έχουμε ένα δεύτερο με την ίδια λογική, στον οποίο διαγράφουμε όλα τα φύλλα εργασίας εκτός από αυτό που περιέχει τα συγκεντρωτικά στοιχεία με τη μέθοδο **Delete** του αντικειμένου **Worksheet**.
- Τέλος αποθηκεύουμε το νέο φύλλο εργασίας που έχει απομείνει με τη μέθοδο **SaveAs**, σε μορφή html αρχείου, δίνοντας στην παράμετρο **FileFormat** την τιμή **xlHtml**. Ως όνομα (τιμή της παραμέτρου **Filename**) χρησιμοποιούμε τη μεταβλητή αλφαριθμητικού **fname**.

```
Sub Create_Html_Report()

  Dim wrksht As Worksheet
  Dim wrksht_id As Integer
  Dim wrksht_count As Integer
  Dim fname As String

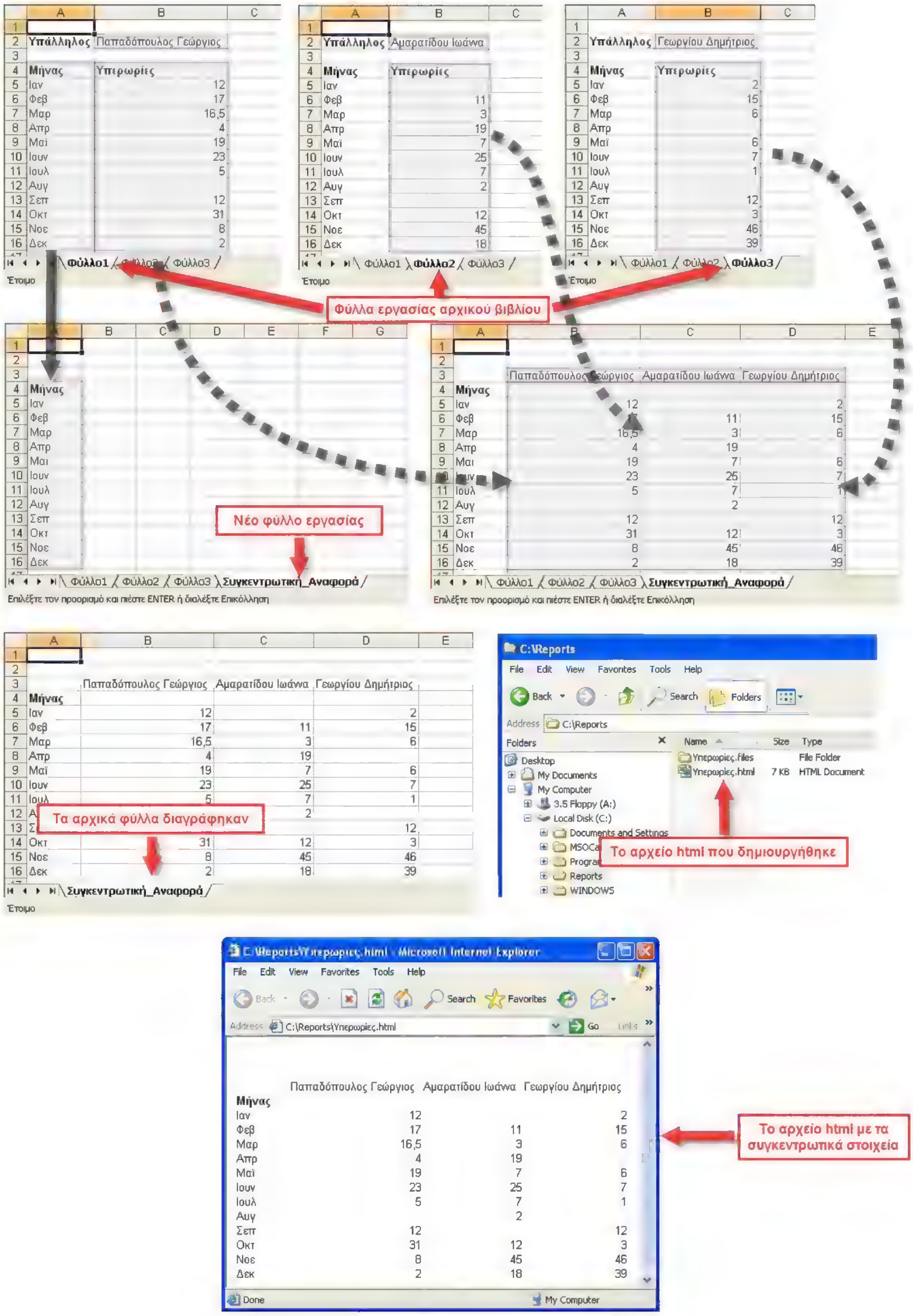
  ThisWorkbook.Activate

  wrksht_count = ThisWorkbook.Worksheets.Count
  ThisWorkbook.Worksheets.Add After:=Worksheets(wrksht_count)
  wrksht_count = ThisWorkbook.Worksheets.Count
  ThisWorkbook.Worksheets(wrksht_count).Name = "Συγκεντρωτική_Αναφορά"
  ThisWorkbook.Worksheets(1).Range("A4:A16").Copy
  ThisWorkbook.Worksheets(wrksht_count).Paste _
  Destination:=Worksheets(wrksht_count).Cells(4, 1)

  For Each wrksht In ThisWorkbook.Worksheets
    wrksht_id = wrksht.Index
    If wrksht_id < ThisWorkbook.Worksheets.Count Then
      wrksht.Range("B2").Copy
      Worksheets("Συγκεντρωτική_Αναφορά").Paste _
      Destination:=Worksheets("Συγκεντρωτική_Αναφορά"). _
      Cells(3, wrksht_id + 1)
      Worksheets("Συγκεντρωτική_Αναφορά").Cells(3, wrksht_id + 1). _
      Columns.AutoFit
      wrksht.Range("B5:B16").Copy
      Worksheets("Συγκεντρωτική_Αναφορά").Paste _
      Destination:=Worksheets("Συγκεντρωτική_Αναφορά"). _
      Cells(5, wrksht_id + 1)
    End If
  Next wrksht

  fname = "C:\Reports\Υπερωρίες.html"
  For Each wrksht In ThisWorkbook.Worksheets
    wrksht_id = wrksht.Index
    If wrksht_id < ThisWorkbook.Worksheets.Count Then
      wrksht.Delete
    End If
  Next wrksht
  Worksheets("Συγκεντρωτική_Αναφορά").Activate
  ActiveSheet.SaveAs Filename:=fname, FileFormat:=xlHtml

End Sub
```



Εικόνα 52 Οθόνες από το προηγούμενο παράδειγμα

Τα συμβάντα του αντικειμένου **Worksheet** είναι τα παρακάτω:

Συμβάν	
Activate	Εκτελεί τον κώδικα όταν ενεργοποιείται το φύλλο εργασίας
BeforeDoubleClick	Εκτελεί τον κώδικα αμέσως πριν από το αποτέλεσμα που θα προκαλέσει το διπλό κλικ του ποντικιού στο φύλλο εργασίας
BeforeRightClick	Εκτελεί τον κώδικα αμέσως πριν από το αποτέλεσμα που θα προκαλέσει το αριστερό κλικ του ποντικιού στο φύλλο εργασίας
Calculate	Εκτελεί τον κώδικα όταν υπολογίζονται οι τιμές των κελιών του φύλλου εργασίας
Change	Εκτελεί τον κώδικα όταν αλλάζει ένα η τιμή ενός ή περισσότερων κελιών
Deactivate	Εκτελεί τον κώδικα όταν απενεργοποιείται το φύλλο εργασίας
FollowHyperlink	Εκτελεί τον κώδικα όταν κάνουμε κλικ σε μία υπερσύνδεση που βρίσκεται στο φύλλο εργασίας
PivotTableUpdate	Εκτελεί τον κώδικα όταν ανανεώνονται τα περιεχόμενα μίας Αναφοράς Συγκεντρωτικού Πίνακα που βρίσκεται στο φύλλο εργασίας
SelectionChange	Εκτελεί τον κώδικα όταν επιλέγουμε μία άλλη περιοχή κελιών του φύλλου εργασίας

Πίνακας 10 Συμβάντα του αντικειμένου Worksheet

Ιδιαίτερα χρήσιμα είναι τα συμβάντα **Activate** και **Deactivate**. Με αυτά μπορούμε να εκτελέσουμε κώδικα όταν ενεργοποιείται ή απενεργοποιείται ένα φύλλο εργασίας.

Συναρτήσεις (Functions)

Οι συναρτήσεις είναι διαδικασίες και περιέχουν κώδικα, όπως και οι υπορουτίνες. Όπως είδαμε, διαφοροποιούνται από αυτές γιατί οι συναρτήσεις επιστρέφουν μία τιμή. Συνήθως εκτελούν κάποιους υπολογισμούς και δέχονται **ορίσματα** (**arguments**) που χρησιμοποιούν για την εκτέλεση των υπολογισμών όπως και οι γνωστές συναρτήσεις του Excel. Μπορούν να καλούνται από άλλες διαδικασίες,. Παράλληλα, μπορούν να δημιουργούνται και συναρτήσεις χρήστη (user functions) στο Excel.

Δήλωση μίας συνάρτησης

Οι συναρτήσεις, όπως και οι υπορουτίνες, δηλώνονται με ένα συγκεκριμένο τρόπο. Η πλήρης σύνταξη έχει τη παρακάτω μορφή:

```
[Public ή Private] [Static] Function function_name ([argument_list]) [As data_type]
        [VBA_instructions]
        [function_name = expression]
        [Exit Function]
        [VBA_instructions]
        [function_name = expression]
End Function
```

όπου:

- **Public**: (προαιρετικά) δηλώνει ότι η συνάρτηση μπορεί να κληθεί από όλες τις διαδικασίες και από οποιοδήποτε module σε όλα τα ενεργά Excel VBA projects
- **Private**: (προαιρετικά) δηλώνει ότι η συνάρτηση μπορεί να κληθεί από όλες τις διαδικασίες που βρίσκονται στο ίδιο μόνο module
- **Static**: (προαιρετικά) δηλώνει ότι οι τιμές των μεταβλητών της συνάρτησης διατηρούνται αναλλοίωτες μεταξύ διαδοχικών κλήσεων της συνάρτησης
- **Function**: (υποχρεωτικά) δεσμευμένη έκφραση που δηλώνει την έναρξη μίας συνάρτησης
- **function_name**: (υποχρεωτικά) το όνομα της συνάρτησης (ακολουθεί τους κανόνες ονομάτων των μεταβλητών)
- **argument_list**: (προαιρετικά) ένα ή περισσότερα ορίσματα (χωρίζονται με κόμμα) που περνάνε στη συνάρτηση. Αν πριν από κάποιο όρισμα γράψουμε **Optional** τότε η χρήση του είναι προαιρετική (όπως και σε συναρτήσεις του Excel που έχουν προαιρετικά ορίσματα)
- **data_type**: (προαιρετικά) ο τύπος δεδομένων που επιστρέφει η συνάρτηση
- **VBA_instructions**: (προαιρετικά) μία ή περισσότερες εντολές της VBA (παρότι είναι προαιρετικές οι εντολές είναι το ουσιαστικό μέρος της συνάρτησης και χωρίς αυτές μία συνάρτηση δεν έχει νόημα)
- **Exit Function**: (προαιρετικά) μία δήλωση που υποχρεώνει τη VBA να σταματήσει την εκτέλεση του κώδικα σε αυτό το σημείο και να επιστρέψει στη διαδικασία που κάλεσε τη συνάρτηση

- **End Function**: (υποχρεωτικά) δηλώνει το τέλος της συνάρτησης. Η VBA επιστρέφει στη διαδικασία που κάλεσε τη συνάρτηση για να συνεχίσει την εκτέλεση του κώδικα

Στην απλούστερη περίπτωση μπορούμε να πληκτρολογήσουμε τη λέξη **Function**, το όνομα της συνάρτησης, και να πατήσουμε Enter, οπότε αυτόματα προστίθεται η εντολή **End Function**, που δηλώνει το τέλος του κώδικα που περιέχει η συνάρτηση. Μετά την γραμμή δήλωσης της συνάρτησης και πριν την εντολή **End Function** εισάγουμε τον κώδικα που επιθυμούμε.

Σημαντική ιδιαιτερότητα των συναρτήσεων είναι ότι σε κάποιο σημείο του κώδικα, πρέπει να αποδώσουμε την τιμή που θέλουμε να επιστρέψει αυτή, σε μία μεταβλητή που έχει το ίδιο όνομα με τη συνάρτηση. Η μεταβλητή έχει τον τύπο δεδομένων που δηλώνεται στη πρώτη γραμμή της συνάρτησης.

Ο κώδικας που ακολουθεί αφορά στη συνάρτηση με όνομα **Add_Cells** και η οποία προσθέτει τα περιεχόμενα των κελιών A1 και B1 και καταχωρεί το αποτέλεσμα στη μεταβλητή Add_Cells που είναι και το όνομα της συνάρτησης.

```
Function Add_Cells(a as Range, b as Range) as Variant
        Add_Cells =a.Value + b.Value
End Function
```

Τρία τμήματα αποτελούν την συνάρτηση **Add_Cells**

1. Η γραμμή δήλωσης της συνάρτησης:

   Function Add_Cells(a as Range, b as Range) as Variant

2. Οι γραμμές που περιέχουν τον κώδικα της υπορουτίνας:

   Add_Cells =a.Value + b.Value

3. Η γραμμή που δηλώνει το τέλος της συνάρτησης:

   End Function

Εκτέλεση συναρτήσεων

Μία υπορουτίνα εκτελείται από το μενού **Run – Run Sub/UserForm**, ή πατώντας **F5.** Μία συνάρτηση δεν μπορεί να εκτελεστεί από μόνη της επειδή πρέπει να επιστρέψει μία τιμή. Επομένως θα πρέπει είτε να κληθεί από μία άλλη διαδικασία (υπορουτίνα ή συνάρτηση) ή μέσα από το περιβάλλον του Excel ως **συνάρτηση χρήστη** (**user function**). Η τελευταία περίπτωση είναι ιδιαίτερα χρήσιμη γιατί παρέχει στο χρήστη τη δυνατότητα να δημιουργεί συναρτήσεις που καλύπτουν τις δικές του ανάγκες και να τις καλεί όποτε απαιτείται.

Κλήση συναρτήσεων από διαδικασίες

Για να καλέσουμε μία συνάρτηση μέσα από μία άλλη διαδικασία θα πρέπει να έχουμε εξασφαλίσει ότι η επιστρεφόμενη τιμή από τη συνάρτηση θα αποδοθεί σε κάποια μεταβλητή ή μία έκφραση η οποία μπορεί να πάρει τιμές, όπως για παράδειγμα είναι οι ιδιότητες των αντικειμένων, ή οι παράμετροι των μεθόδων.

Στο επόμενο παράδειγμα πραγματοποιείται η κλήση της συνάρτησης **Add_Cells** μέσα από την υπορουτίνα **Call_Function**:

```
Sub Call_function()

   Dim Result As Variant
   Result = Add_Cells(Cells(1, 1), Cells(1, 2))
   MsgBox Result

End Sub
```

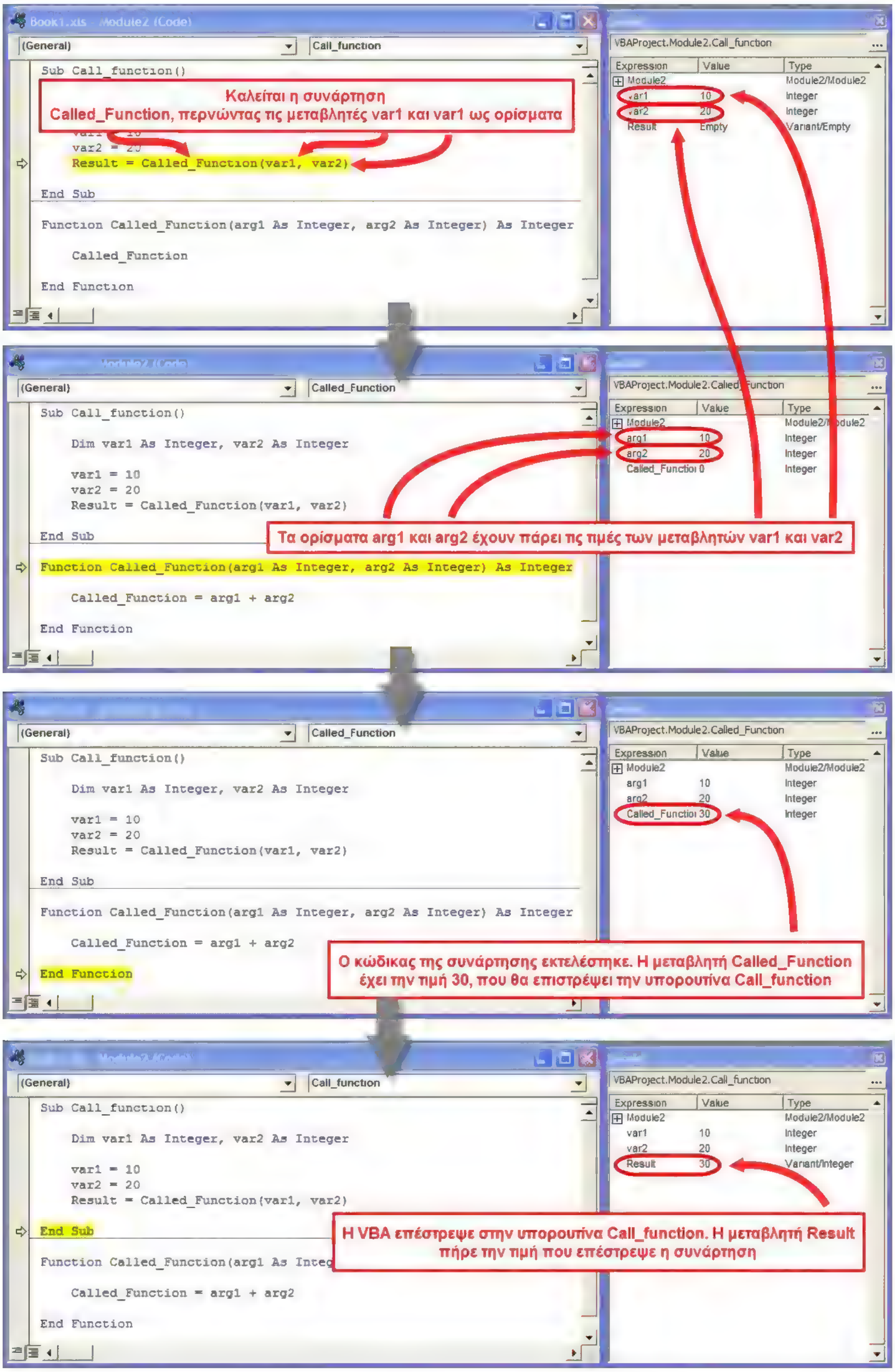
```
Function Add_Cells(a As Range, b As Range) As Variant

   Add_Cells = a.Value + b.Value

End Function
```

Στη συνάρτηση περνάμε δύο ορίσματα, το **Cell(1,1)** και το **Cell(1,2)**, δηλαδή δύο αντικείμενα **Range**. Στη δήλωση της συνάρτησης, καθορίζουμε ότι δέχεται δύο ορίσματα, τα **a** και **b** που είναι αντικείμενα **Range**. Όταν καλείται η συνάρτηση το **a** παίρνει τη τιμή που έχει το αντικείμενο **Cell(1,1)** και το **b** τη τιμή που έχει το αντικείμενο **Cell(1,1)**. Η τιμή που υπολογίζει η συνάρτηση **Add_Cells**, αποδίδεται στη μεταβλητή **Result** της υπορουτίνας **Call_Function**.

Παρακάτω απεικονίζεται η κλήση μίας συνάρτησης από μία υπορουτίνα, και πως μπορούμε να παρακολουθούμε στο Locals Window τις τιμές των μεταβλητών και των ορισμάτων.



Εικόνα 53 Κλήση συνάρτησης από μία διαδικασία

Επίσης όπως και με τις υπορουτίνες μπορούμε να χρησιμοποιήσουμε τη μέθοδο **Run** του αντικειμένου **Application** ως εξής:

```
Sub Call_function()

   Dim var1 As Integer, var2 As Integer

   var1 = 10
   var2 = 20
   Result = Application.Run("Called_Function", var1, var2)

End Sub
```

```
Function Called_Function(arg1 As Integer, arg2 As Integer) As Integer

   Called_Function = arg1 + arg2

End Function
```

Κλήση συναρτήσεων από φύλλο εργασίας

Οι συναρτήσεις που δημιουργούμε στη VBA, μπορούν να χρησιμοποιηθούν όπως οι συνηθισμένες συναρτήσεις του Excel (SUM, AVERAGE, IF κλπ), στα φύλλα εργασίας ενός βιβλίου. Αν η συνάρτηση περιέχεται στο βιβλίο όπου θέλουμε να τη χρησιμοποιήσουμε, τότε έχουμε πρόσβαση σε αυτή όπως στις υπόλοιπες συναρτήσεις. Μπορούμε δηλαδή σε ένα κελί να πληκτρολογήσουμε απευθείας το όνομα της συνάρτησης και να δώσουμε τα ανάλογα ορίσματα. Η συνάρτηση μας επίσης εμφανίζεται στο παράθυρο **Εισαγωγή συνάρτησης** στην κατηγορία **Συναρτήσεις χρήστη**, όπου θα βλέπουμε κα τα ορίσματα που πρέπει να δώσουμε. Αν είναι ανοικτό το βιβλίο που περιέχει τη συνάρτηση μας τότε η συνάρτηση είναι ορατή και στο παράθυρο **Εισαγωγή συνάρτησης** και σε κάθε άλλο βιβλίο που έχουμε ταυτόχρονα ανοικτό.

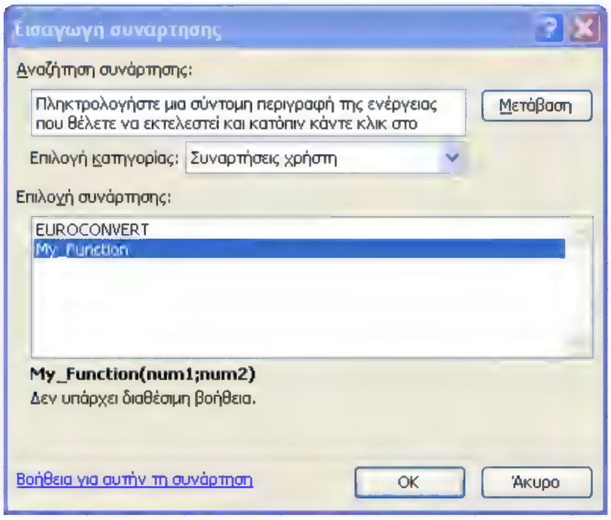
Αν το βιβλίο που περιέχει τη συνάρτηση δεν είναι ανοικτό, τότε για να χρησιμοποιήσουμε τη συνάρτηση μας πρέπει να ορίσουμε μία αναφορά στο βιβλίο που περιέχεται η συνάρτηση από το μενού **Tool – Reference** του Επεξεργαστή VBA .

Η συνάρτηση στο επόμενο παράδειγμα υπολογίζει την τετραγωνική ρίζα του αθροίσματος των τετραγώνων δύο αριθμών:

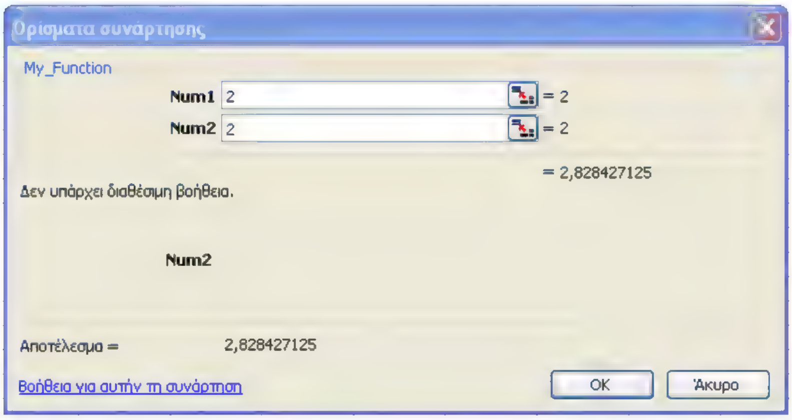
```
Function My_Function(num1 As Double, num2 As Double)

   My_Function = Sqr((num1) ^ 2 + (num2) ^ 2)

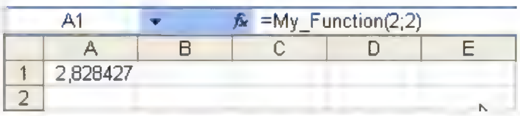
End Function
```



Εικόνα 54 Η συνάρτηση μας εμφανίζεται στο παράθυρο Εισαγωγή συνάρτησης



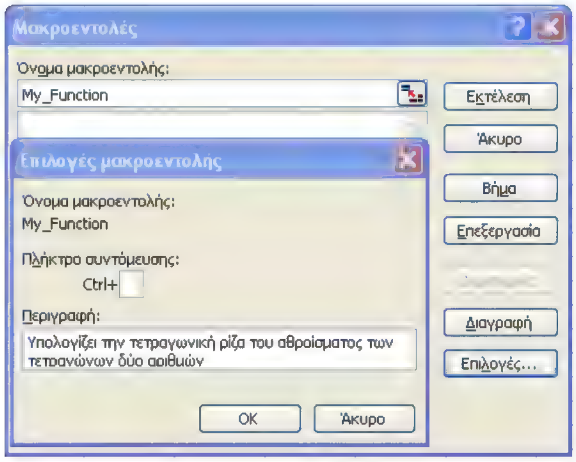
Εικόνα 55 Εισαγωγή ορισμάτων στη συνάρτηση μας

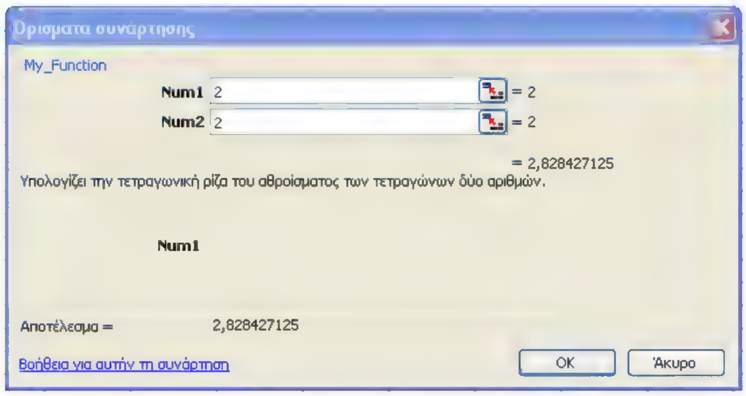


Εικόνα 56 Η συνάρτηση μας στο φύλλο εργασίας

Αν θέλουμε να εμφανίζεται περιγραφή της συνάρτησης μας στο παράθυρο **Εισαγωγή συνάρτησης** ακολουθούμε τα παρακάτω βήματα:

- Από τη διαδρομή του μενού **Εργαλεία – Μακροεντολή – Μακροεντολές**, εμφανίζουμε το σχετικό παράθυρο και στο πεδίο **Όνομα μακροεντολής** πληκτρολογούμε το όνομα της συνάρτησης που έχουμε δημιουργήσει (εδώ εμφανίζονται αυτόματα μόνο υπορουτίνες και όχι συναρτήσεις)
- Πατάμε το κουμπί **Επιλογές**
- Στο παράθυρο που εμφανίζεται εισάγουμε την περιγραφή που επιθυμούμε στο σχετικό πεδίο
- Πατάμε **OK**
- Στο παράθυρο **Μακροεντολές** πατάμε **Άκυρο**





Κλήση συναρτήσεων του Excel στη VBA

Σχεδόν όλες οι ένθετες συναρτήσεις του Excel είναι δυνατόν να εισαχθούν σε κώδικα της VBA, χρησιμοποιώντας τη μέθοδο **WorksheetFunction** του αντικειμένου Application, όπως παρακάτω:

```
Sub Excel_function()

   Dim rng As Range
   Set rng = Range("A1:A4")
   Result = Application.WorksheetFunction.Sum(rng)

End Sub
```

Στο παράδειγμα χρησιμοποιούμε τη συνάρτηση **Sum** του Excel για να αθροίσουμε τη περιοχή κελιών A1:A4. Ως όρισμα για τη **Sum** εισάγουμε τη περιοχή **rng** που έχουμε δηλώσει προηγουμένως και της έχουμε αποδώσει τιμή με την εντολή **Set rng = Range("A1:A4")**.

Ένθετες συναρτήσεις της VBA

Η VBA μας παρέχει ένα σχετικά περιορισμένο αριθμό συναρτήσεων, όπως για παράδειγμα η MsgBox που έχουμε δει. Οι συναρτήσεις κατηγοριοποιουνται κατά λειτουργικότητα ως εξής:

- Συναρτήσεις πινάκων (arrays)
- Μετατροπής δεδομένων
- Ημερομηνίας και ώρας
- Διαχείρισης αρχείων και φακέλων
- Οικονομικές συναρτήσεις
- Συναρτήσεις πληροφορίας
- Μαθηματικές
- Διαχείρισης αλφαριθμητικών

Έλεγχος εκτέλεσης κώδικα

Λήψη αποφάσεων

Μέσω της VBA μπορούμε να δημιουργήσουμε κώδικα που να μας παρέχει τη δυνατότητα να εκτελούμε κάποιο τμήμα του κώδικα υπό συνθήκες κα να είναι εφικτή η λήψη αποφάσεων. Έτσι, μπορούμε για παράδειγμα να εκτελούμε συγκεκριμένες εντολές αν μία συνθήκη είναι αληθής, και διαφορετικές εντολές αν είναι ψευδής. Για να εκτελούμε κώδικα υπό συνθήκες χρησιμοποιούμε δύο εντολές της VBA, η πρώτη είναι η εντολή **IF** και μοιάζει με τη γνωστή συνάρτηση IF του Excel, και η δεύτερη είναι η **Select**.

Η εντολή IF

Η πλέον συνηθισμένη σύνταξη της εντολής IF είναι η παρακάτω:

```
If condition Then
        [true_VBA_instructions]
[ElseIf condition Then
        [false_VBA_instruction]]
[Else
        [default_VBA_instructions]]
End If
```

όπου:

- **If**: (υποχρεωτικά) δηλώνει την αρχή της εντολής **If**
- **condition**: (υποχρεωτικά) η συνθήκη που ελέγχεται και πρέπει να είναι μία έκφραση που επιστρέφει σαν αποτέλεσμα True ή False. Στη πρώτη περίπτωση εκτελούνται οι εντολές **true_VBA_instructions**, ενώ στη δεύτερη όχι
- **Then**: (υποχρεωτικά) έκφραση που δηλώνει ότι έπονται οι εντολές που θα εκτελεστούν εφόσον η συνθήκη είναι αληθής
- **true_VBA_instructions**: (προαιρετικά) μία ή περισσότερες εντολές VBA που θα εκτελεστούν, εφόσον η συνθήκη είναι αληθής (παρότι είναι προαιρετικές, οι εντολές είναι το ουσιαστικό μέρος της εντολής)
- **ElseIf**: (προαιρετικά) έκφραση που δηλώνει ότι έπονται οι εντολές που θα εκτελεστούν εφόσον η προηγούμενη συνθήκη είναι ψευδής και η συνθήκη που ακολουθεί αληθής
- **condition**: (προαιρετικά) συνθήκη που ελέγχεται
- **false_VBA_instructions**: (προαιρετικά) μία ή περισσότερες εντολές VBA που θα εκτελεστούν, εφόσον η προηγούμενη συνθήκη είναι αληθής
- **Else**: (προαιρετικά) έκφραση που δηλώνει ότι οι εντολές που έπονται θα εκτελεστούν εφόσον όλες οι προηγούμενες συνθήκες είναι ψευδείς
- **Default_VBA_instructions**: (προαιρετικά) μία ή περισσότερες εντολές VBA που θα εκτελεστούν εφόσον όλες οι προηγούμενες συνθήκες είναι ψευδείς
- **End If**: (υποχρεωτικά) έκφραση που δηλώνει το τέλος της εντολής

Η εντολή Select

Στην περίπτωση όπου έχουμε μία συνθήκη που μπορεί να πάρει πολλές διαφορετικές τιμές, και επιθυμούμε να εκτελούμε για κάθε τιμή και διαφορετικό κώδικα, χρησιμοποιούμε την εντολή **Select**. Η σύνταξή της είναι η παρακάτω:

```
Select Case expression
        [Case value
                VBA_instructions]
        [Case Else
                VBA_instructions]
End Select
```

όπου:

- **Select Case**: (υποχρεωτικά) δηλώνει την αρχή της εντολής
- **expression**: (υποχρεωτικά) η έκφραση, της οποίας η τιμή θα ελέγχει την επιλεκτική εκτέλεση τμημάτων του κώδικα που ακολουθούν
- **Case**: (προαιρετικά) δηλώνει ότι ακολουθεί η τιμή με την οποία θα συγκριθεί η τιμή της έκφρασης
- **value**: (προαιρετικά) η τιμή με την οποία αν συμφωνεί η τιμή της μεταβλητής θα εκτελεστούν οι εντολές που ακολουθούν
- **VBA_instructions**: (προαιρετικά) εντολές που θα εκτελεστούν. Συνήθως υπάρχουν περισσότερα από ένα ζεύγη Case value – VBA_instructions
- **Case Else**: (προαιρετικά) έκφραση που δηλώνει ότι οι επόμενες εντολές θα εκτελεστούν εφόσον η τιμή της έκφρασης δεν συμφωνεί με καμία από τις προηγούμενες τιμές
- **End Select**: (υποχρεωτικά) Δηλώνει το τέλος της εντολής

Ελεγχόμενη επανάληψη εκτέλεσης κώδικα

Πολλές φορές απαιτείται να εκτελεστεί ο ίδιος κώδικας πολλές φορές, αλλάζοντας συνήθως μόνο τιμές κάποιων μεταβλητών. Για παράδειγμα αν θέλουμε να ελέγξουμε στην περιοχή A1:A10 αν οι τιμές τους ισούνται με μία συγκεκριμένη τιμή z, θα πρέπει να εκτελέσουμε μία εντολή όπως Cells(x, y) = z, όπου το y θα παίρνει τιμές από 1 έως 10, έτσι ώστε να ελεγχθούν και τα 10 κελιά. Για να μην γράφουμε πολλές φορές τις ίδιες εντολές χρησιμοποιούμε ένα βρόχο, που περιέχει μία φορά τις εντολές, και η εκτέλεση του επαναλαμβάνεται ελεγχόμενα όσες φορές απαιτείται. Στις εντολές που περιέχει ο βρόχος χρησιμοποιούμε συνήθως μεταβλητές, των οποίων οι τιμές μεταβάλλονται ώστε σε κάθε εκτέλεση να επιτυγχάνουμε το επιθυμητό αποτέλεσμα.

Η εντολή For

Η εντολή For επιτρέπει την εκτέλεση εντολών για συγκεκριμένο αριθμό επαναλήψεων. Χρησιμοποιεί ένα μετρητή που αυξάνει κατά συγκεκριμένη τιμή κάθε φορά που εκτελείται ο βρόχος και η ελέγχεται σε σχέση με μία τελική τιμή. Οι επαναλήψεις εκτελούνται όσο η τιμή του μετρητή είναι μικρότερη ή ίση με την τελική τιμή. Η σύνταξη της είναι η παρακάτω:

```
For counter = start To stop [Step increment]
        [VBA_instructions]
Next counter
```

όπου:

- **For**: (υποχρεωτικά) δηλώνει την αρχή του βρόχου
- **counter**: (υποχρεωτικά) μεταβλητή αριθμητικού τύπου που εκτελεί χρέη μετρητή των επαναλήψεων που έχουν εκτελεστεί
- **start**: (υποχρεωτικά) η αρχική τιμή που λαμβάνει η counter
- **To**: (υποχρεωτικά) δηλώνει ότι η τιμή που ακολουθεί είναι η τελική τιμή που θα λάβει η counter
- **stop**: (υποχρεωτικά) η τελική τιμή που λαμβάνει η counter
- **Step**: (προαιρετικά) στην περίπτωση όπου η τιμή του μετρητή δεν αυξάνει κατά 1 σε κάθε επανάληψη, δηλώνει ότι η τιμή που ακολουθεί είναι το βήμα για το μετρητή
- **increment**: (προαιρετικά) η τιμή του βήματος στην περίπτωση που δεν είναι ίσο με 1
- **VBA_instructions**: (προαιρετικά) οι εντολές που εκτελούνται σε κάθε επανάληψη και αποτελούν το κορμό του βρόχου
- **Next counter**: (υποχρεωτικά) δηλώνει το τέλος του βρόχου

Το επόμενο παράδειγμα υπολογίζει τη μέση βαθμολογία 10 μαθημάτων και εμφανίζει ένα πλαίσιο διαλόγου με το αποτέλεσμα. Οι βαθμολογίες είναι καταχωρημένες στα κελιά A1 έως A10. Χρησιμοποιούμε την εντολή **For** για να υπολογίσουμε το άθροισμα των επιμέρους βαθμολογιών, εντολή **If**, για να εμφανίζουμε διαφορετικό μήνυμα ανάλογα με το αποτέλεσμα και την εντολή:

```
Sub If_demo()

  Dim N As Integer
  Dim Average As Single, Sum As Single

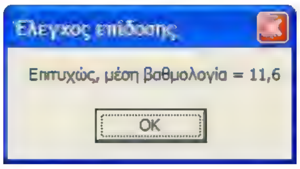
  For N = 1 To 10
    Sum = Sum + Cells(N, 1)
  Next N
  Average = Sum / (N - 1)

  If Average >= 10 And Average <= 20 Then
    MsgBox Prompt:="Επιτυχώς, μέση βαθμολογία = " & Average, _
    Title:="Έλεγχος επίδοσης"
  ElseIf Average < 10 And Average > 0 Then
    MsgBox Prompt:="Ανεπιτυχώς, μέση βαθμολογία = " & Average, _
    Title:="Έλεγχος επίδοσης"
  Else
```

```
    MsgBox Prompt:="Λάθος βαθμολογία", Title:="Έλεγχος επίδοσης"
  End If

End Sub
```

Το πλαίσιο διαλόγου εμφανίζει το παρακάτω μήνυμα, όταν η μέση βαθμολογία είναι μεγαλύτερη ή ίση του 10:



Εικόνα 57

Η εντολή For Each

Στη περίπτωση όπου έχουμε μία συλλογή και θέλουμε να εκτελέσουμε μία ή περισσότερες εντολές για κάθε αντικείμενο τα συλλογής, χρησιμοποιούμε την εντολή For Each. Η σύνταξη της είναι η παρακάτω:

```
For Each item in collection
      [VBA_instructions]
      [Exit For]
      [VBA_instructions]
Next item
```

όπου:

- **For**: (υποχρεωτικά) δηλώνει την αρχή του βρόχου
- **item**: (υποχρεωτικά) μεταβλητή αντικειμένου που ανήκει στη συλλογή
- **VBA_instructions**: (προαιρετικά) οι εντολές που εκτελούνται σε κάθε επανάληψη και αποτελούν το κορμό του βρόχου
- **Exit For**: (προαιρετικά) δηλώνει ότι σταματάει η εκτέλεση του βρόχου (συνήθως έπεται μίας εντολής που ελέγχει μία συνθήκη, όπως η **If**)
- **Next item**: (υποχρεωτικά) δηλώνει το τέλος του βρόχου

Στο επόμενο παράδειγμα ο βρόχος **For Each … Next** εκτυπώνει όλα τα φύλλα εργασίας του ενεργού βιβλίου. Η **wrksht** δηλώνεται ως μεταβλητή αντικειμένου **Worksheet**.

```
Sub Print_All_Worksheets()

  Dim wrksht As Worksheet
    For Each wrksht In ActiveWorkbook.Worksheets
    wrksht.PrintOut
  Next wrksht

End Sub
```

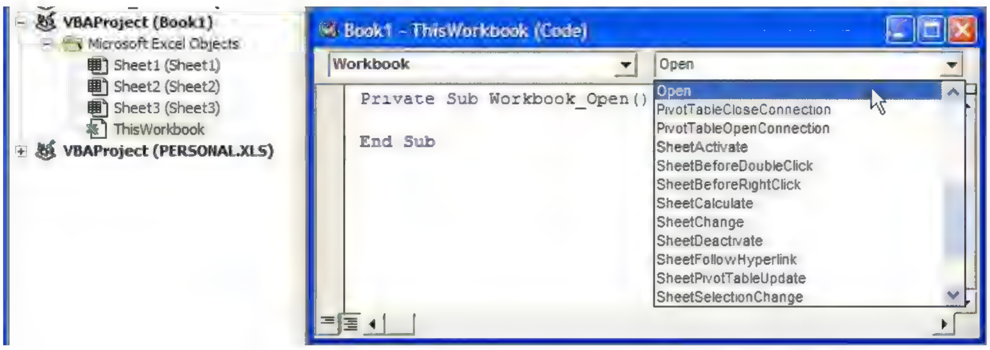
Συμβάντα

Όταν ανοίγουμε ένα βιβλίο εργασίας έχουμε μία συγκεκριμένη ενέργεια που λαμβάνει χώρα, με άλλα λόγια έχουμε ένα **συμβάν** (**event**). Το Excel ανιχνεύει τα συμβάντα και ανταποκρίνεται αναλόγως σε αυτά. Επίσης μπορεί να εκτελεί προκαθορισμένες διαδικασίες της VBA όταν λαμβάνουν χώρα συγκεκριμένα συμβάντα. Η VBA χρησιμοποιεί **διαδικασίες διαχείρισης συμβάντων** (**event handler procedures**) για να ανταποκρίνεται στα συμβάντα και οι οποίες έχουν συγκεκριμένα ονόματα. Τα συμβάντα που είναι σε θέση να παρακολουθεί το Excel, κατά κανόνα συνδέονται με αντικείμενα και κατηγοριοποιούνται ως εξής:

- Συμβάντα βιβλίου εργασίας (workbook events): αφορούν ενέργειες που πραγματοποιούνται σε συγκεκριμένο βιβλίο εργασίας, όπως **Open** όταν ανοίγει το βιβλίο, **BeforeSave**, όταν αποθηκεύουμε το βιβλίο ή **NewSheet** όταν προσθέτουμε νέο φύλλο στο βιβλίο
- Συμβάντα φύλλου εργασίας (worksheet events): αφορούν ενέργειες που πραγματοποιούνται σε συγκεκριμένο φύλλο εργασίας, όπως **Change** όταν αλλάζει ένα κελί του φύλλου
- Συμβάντα γραφήματος (chart events): αφορούν ενέργειες που πραγματοποιούνται σε συγκεκριμένο γράφημα, όπως **Select** όταν επιλέγεται ένα αντικείμενο του γραφήματος
- Συμβάντα εφαρμογής (application events): αφορούν ενέργειες σε επίπεδο εφαρμογής του Excel όπως **NewWorkbook** όταν δημιουργούμε ένα νέο βιβλίο εργασίας
- Συμβάντα φορμών χρήστη (user forms event): αφορούν ενέργειες πραγματοποιούνται σε συγκεκριμένη φόρμα χρήση, όπως Click όταν πατάμε ένα κουμπί στη φόρμα
- Συμβάντα που δεν συνδέονται άμεσα με αντικείμενο: όπως **OnKey** όταν πατάμε συγκεκριμένο συνδυασμό πλήκτρων στο πληκτρολόγιο

Που αποθηκεύονται οι διαδικασίες διαχείρισης συμβάντων

Οι διαδικασίες που εκτελούνται όταν αναγνωρίζεται ένα συγκεκριμένο συμβάν δεν αποθηκεύονται στον κώδικα ενός module που δημιουργήσαμε εμείς, αλλά στον κώδικα του αντικειμένου με το οποίο σχετίζονται. Επίσης έχουν συγκεκριμένα ονόματα, βάσει των οποίων συνδέονται με συγκεκριμένα συμβάντα και εκτελούνται όταν αυτά ανιχνεύονται.



Εικόνα 58 Εισαγωγή της διαδικασίας Workbook_Open

Για να εισάγουμε στο κώδικα ενός αντικειμένου μία διαδικασία διαχείρισης συμβάντος, επιλέγουμε το αντικείμενο στο παράθυρο του **Project Browser** με διπλό κλικ, και στο παράθυρο κώδικα που ανοίγει

επιλέγουμε στο **Procedure list box** το συμβάν που θέλουμε. Αυτόματα δημιουργείται η δήλωση της διαδικασίας.

Το επόμενο παράδειγμα αποθηκεύει ένα αντίγραφο του βιβλίου εργασίας, μόλις το ανοίξουμε:

```
Private Sub Workbook_Open()
   Dim dt As Variant

   dt = Date
   dt = Replace(dt, "/", "-")
   ActiveWorkbook.SaveCopyAs "C:\VBA Files\Backup " & dt & ".xls"

End Sub
```

Χρησιμοποιούμε τις συναρτήσεις **Date** και **Replace** της VBA. Η πρώτη επιστρέφει την τρέχουσα ημερομηνία και την αποθηκεύει στη μεταβλητή **dt**, ενώ η δεύτερη αντικαθιστά το χαρακτήρα **"/"** με το χαρακτήρα **"-"**. Στη συνέχεια με τη μέθοδο **SaveAsCopy** αποθηκεύουμε ένα αντίγραφο του βιβλίου, χρησιμοποιώντας τη μεταβλητή **dt** ως μέρος του ονόματος του αρχείου.

Στο επόμενο παράδειγμα χρησιμοποιούμε το συμβάν **Worksheet_SelectionChange**, που προκύπτει όταν επιλέξουμε μία διαφορετική περιοχή κελιών. Η διαδικασία αποθηκεύεται στο κώδικα του φύλλου εργασίας, και καλείται μόνο όταν επιλέξουμε μία νέα περιοχή σε αυτό.

```
Private Sub Worksheet_SelectionChange(ByVal Target As Excel.Range)
' Η διαδικασία καλείται όταν επιλέξουμε μία νέα περιοχή κελιών

   ' Η έκφραση Cells επιστρέφει όλα τα κελιά του ενεργού φύλλου
   ' Μορφοποιούμε τα όλα κελιά χωρίς χρώμα γεμίσματος
   Cells.Interior.ColorIndex = xlNone

   With ActiveCell
      .EntireRow.Interior.ColorIndex = 24
      .EntireColumn.Interior.ColorIndex = 24
   End With

End Sub
```

Αλληλεπίδραση με το χρήστη

Στο παραθυρικό περιβάλλον εργασίας των Windows η αλληλεπίδραση μεταξύ χρηστών και εφαρμογών γίνεται σε μαγάλο βαθμό μέσω **πλαισίων διαλόγου** (**dialog boxes**). Η VBA παρέχει τη δυνατότητες δημιουργίας και διαχείρισης πλαισίων διαλόγου. Αυτά μπορεί να είναι προκαθορισμένα ή να σχεδιάζονται από το χρήστη τόσο ως προς την εμφάνιση, όσο και προς τη λειτουργικότητα τους. Τα τελευταία ονομάζονται **φόρμες χρήστη** (**User Forms**).

Πλαίσια διαλόγου του Excel

Η συνάρτηση InputBox

Η συνάρτηση InputBox της VBA εμφανίζει ένα πλαίσιο διαλόγου όπου ο χρήστης μπορεί να εισάγει δεδομένα, όπως κείμενο, έναν αριθμό ή μία περιοχή κελιών, για να χρησιμοποιηθούν σε κώδικα. Η σύνταξη της συνάρτησης είναι η παρακάτω:

InputBox (prompt[, title][, default][, xpos][, ypos][, helpfile, context])

όπου:

- **prompt**: (υποχρεωτικά) το κείμενο που θα εμφανίζεται πλαίσιο διαλόγου
- **title**: (προαιρετικά) ο τίτλος του πλαισίου διαλόγου
- **default**: (προαιρετικά) προκαθορισμένη τιμή που θα εμφανίζεται στο πλαίσιο διαλόγου (αν ο χρήστης δεν εισάγει άλλη τιμή, η συνάρτηση θα επιστρέψει αυτή την τιμή)
- **xpos, ypos**: (προαιρετικά) η θέση στην οθόνη της άνω αριστερής γωνίας του παλισίου
- **helpfile, context**: (προαιρετικά) αρχείο βοήθειας και το θέμα βοήθειας

Η συνάρτηση MsgBox

Η συνάρτηση **MsgBox** μπορεί να εμφανίζει ένα **μήνυμα** (**message**) σε ένα πλαίσιο διαλόγου, και επίσης να δέχεται την απόκριση του χρήστη όταν αυτός πατάει κουμπιά που εμφανίζονται στο παράθυρο του μηνύματος. Η σύνταξη της συνάρτησης είναι η παρακάτω:

MsgBox (prompt[, buttons][, title][, helpfile, context])

όπου:

- **prompt**: (υποχρεωτικά) το κείμενο που θα εμφανίζεται πλαίσιο διαλόγου
- **buttons**: (προαιρετικά) ένας αριθμός που δηλώνει ποιά κουμπιά και εικονίδια θα εμφανιστούν στο πλαίσιο
- **title**: (προαιρετικά) ο τίτλος του πλαισίου διαλόγου
- **helpfile, context**: (προαιρετικά) αρχείο βοήθειας και το θέμα βοήθειας

Σταθερά	**Τιμή**	**Περιγραφή**
vbOKOnly	0	Εμφανίζεται μόνο το κουμπί OK
vbOKCancel	1	Εμφανίζονται τα κουμπιά Ok και Cancel
vbAbortRetryIgnore	2	Εμφανίζονται τα κουμπιά Abort, Retry και Ignore
vbYesNoCancel	3	Εμφανίζονται τα κουμπιά Yes, No και Cancel
vbYesNo	4	Εμφανίζονται τα κουμπιά Yes και No
vbRetryCancel	5	Εμφανίζονται τα κουμπιά Retry και Cancel
vbCritical	16	Εμφανίζεται το εικονίδιο Critical Message
vbQuestion	32	Εμφανίζεται το εικονίδιο Question Message
vbExlamation	48	Εμφανίζεται το εικονίδιο Exlamation Message
vbInformation	64	Εμφανίζεται το εικονίδιο Information Message
vbDefaultButton1	128	Το πρώτο κουμπί είναι προκαθορισμένο
vbDefaultButton2	256	Το δεύτερο κουμπί είναι προκαθορισμένο
vbDefaultButton3	512	Το τρίτο κουμπί είναι προκαθορισμένο
vbDefaultButton4	768	Το τέταρτο κουμπί είναι προκαθορισμένο
vbSystemModal	4096	Όλες οι εφαρμογές αναστέλλονται μέχρι να αποκριθεί ο χρήστης και να κλείσει το πλαίσιο διαλόγου

Πίνακας 11 Σταθερές που δηλώνουν ποιο/ποια κουμπιά θα εμφανιστούν στο πλαίσιο διαλόγου της συνάρτησης MsgBox

Σταθερά	**Τιμή**
vbOK	1
vbCancel	2
vbAbort	3
vbRetry	4
vbIgnore	5
vbYes	6
vbNo	7

Πίνακας 12 Σταθερές που επιστρέφει η συνάρτηση MsgBox

Στο επόμενο παράδειγμα χρησιμοποιούμε τη συνάρτηση **InputBox** για εισάγει ο χρήστης το όνομα και το επίθετο του. Η συνάρτηση επιστρέφει ότι εισάγει ο χρήστης, στη μεταβλητή **FullName** που είναι τύπου **string**. Στη συνέχεια αναζητούμε το κενό που χωρίζει το όνομα από το επίθετο και τα αποθηκεύουμε στις μεταβλητές **FirstName** και **LastName** αντίστοιχα. Τέλος τα εμφανίζουμε με τη συνάρτηση **MsgBox**.

```
Sub ShowName()

    Dim FullName As String
    Dim FirstName As String
    Dim LastName As String
    Dim FirstSpace As Integer
    Dim Response As Long
```

```
Start:
   FirstName = ""
   LastName = ""
   FirstSpace = 0
   FullName = GetName
   FirstSpace = InStr(FullName, " ")
   If FirstSpace <> 0 Then
      FirstName = Left(FullName, FirstSpace - 1)
      LastName = Right(FullName, Len(FullName) - FirstSpace)
   ElseIf FirstSpace = 0 Then
      FirstName = FullName
      LastName = ""
   End If

   Response = MsgBox"Όνομα χρήστη: " & FirstName, 4, "Είναι σωστά
τα _ παρακάτω στοιχεία;")
   If Response = 7 Then
      GoTo Start
   End If
   Response = MsgBox("Επίθετο χρήστη: " & LastName, 4, " Είναι
σωστά τα _ παρακάτω στοιχεία;")
   If Response = 7 Then
      GoTo Start
   End If

End Sub
```

```
Function GetName() As String

   Do Until GetName <> ""
      GetName = InputBox("Εισάγετε το όνομα και το επίθετό σας", _
      "Στοιχεία Χρήστη")
   Loop

End Function
```

Η συνάρτηση **InputBox** βρίσκεται μέσα σε ένα βρόχο **Do Until**, που επαναλαμβάνεται εφόσον ο χρήστης δεν εισάγει τίποτα στο πλαίσιο διαλόγου (δηλαδή η μεταβλητή **GetName** είναι ίση με ""). Η συνάρτηση InStr αναζητά το χαρακτήρα " ", δηλαδή το κενό που χωρίζει το όνομα από το επίθετο και αποθηκεύει τη θέση του μέσα στο κείμενο στη μεταβλητή **FirstSpace** σαν ένα αριθμό. Οι συνάρτησεις **Left** και **Right** επιστρέφουν το αριστερό ή το δεξί τμήμα από μία μεταβλητή τύπου **string** για όσους χαρακτήρες δηλώνουμε στο δεύτερο όρισμα τους. Η συνάρτηση **Len** επιστρέφει μία τιμή τύπου **Long** που δηλώνει τον αριθμό χαρακτήρων που έχει μία μεταβλητή τύπου **string**.

Η μέθοδος GetOpenFileName

Στην περίπτωση όπου θέλουμε να επιλέξουμε ένα αρχείο, μπορούμε να χρησιμοποιήσουμε τη μέθοδο **GetOpenFileName** του αντικειμένου **Application**. Η μέθοδος εμφανίζει ένα πλαίσιο διαλόγου αντίστοιχο του πλαισίου **Άνοιγμα** του Excel. Δεν ανοίγει όμως ένα αρχείο, αλλά επιστρέφει ένα αλφαριθμητικό που περιέχει τη διαδρομή και το όνομα του αρχείου. Η σύνταξη της είναι η παρακάτω:

Application.GetOpenFileName([filefilter, filteriIndex, title, buttontext, multiselect])

όπου:

- **filefilter**: (προαιρετικά) κριτήριο (**string**) που καθορίζει τα αρχεία που μπορούμε να επιλέξουμε βάσει της επέκτασής τους
- **filterindex**: (προαιρετικά) δείκτης που καθορίζει προτεινόμενο φίλτρου
- **title**: (προαιρετικά) ο τίτλος του πλαισίου διαλόγου
- **buttontext**: (προαιρετικά) μόνο για την Macintosh έκδοση του Excel
- **multiselect**: (προαιρετικά) αν έχει τιμή True τότε μπορούμε να επιλέξουμε περισσότερα από ένα αρχεία

Στο επόμενο παράδειγμα εμφανίζουμε ένα πλαίσιο διαλόγου για να επιλέξουμε ένα ή περισσότερα αρχεία, έχοντας καθορίσει τρεις τύπους αρχείων και επιβεβαι:

```
Sub GetFileName()
   Dim Filter As String
   Dim FilterIndex As Integer
   Dim FileName As Variant
   Dim Title As String

'  Καθορίζουμε τους τύπους αρχείων που μπορούμε να επιλέξπουμε
   Filter = "Excel Files (*.xls),*.xls," & _
        "Comma Separated Files (*.csv),*.csv," & _
        "All Files (*.*),*.*"

'  Προκαθορίζουμε *.* ως κριτήριο επιλογής
   FilterIndex = 3

   Title = "Επιλέξτε αρχεία"

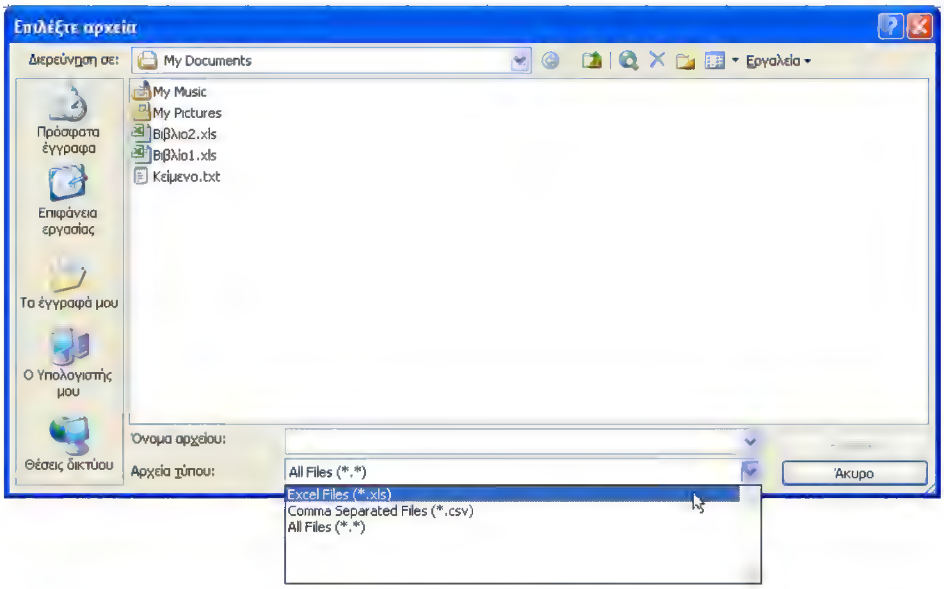
   FileName = Application.GetOpenFilename _
      (FileFilter:=Filter, _
       FilterIndex:=FilterIndex, _
       Title:=Title, _
       MultiSelect:=True)

'  Ελέγχουμε αν έχει επιλεχθεί τουλάχιστον ένα αρχείο
   If Not IsArray(FileName) Then
      MsgBox "Δεν επιλέξατε κάποιο αρχείο"
      Exit Sub
  End If

'  Εμφανίζουμε τα αρχεία που επιλέξαμε
   For i = LBound(FileName) To UBound(FileName)
      Msg = Msg & FileName(i) & vbCrLf
```

```
        Next i
        MsgBox "Επιλέξατε τα παρακάτω αρχεία:" & vbCrLf & Msg

    End Sub
```



Εικόνα 59 Το πλαίσιο διαλόγου που εμφανίζει η μέθοδος GetOpenFileName

Όταν δηλώνουμε **MultiSelect:=True**, επιστέφεται ένα **array**. Αν πατηθεί το κουμπί Cancel το array είναι κενό. Οι συναρτήςσεις **LBound** και **UBound** επιστρέφουν τα κάτω και άνω όρια ενός πίνακα (**array**) αντίστοιχα.

Η μέθοδος GetSaveAsFilename

Η μέθοδος αυτή εμφανίζει ένα πλαίσιο διαλόγου αντίστοιχο του παλισίου **Αποθήκευση ως** του Excel. και επιτρέπει στο χρήστη να επιλέξει ένα αρχείο ή να καθορίσει το όνομα του αρχείου. Η σύνταξη της είναι:

Application.GetSaveAsFileName ([initialfilename, filefilter, title, filterindex, buttontext])

όπου:

- **initialfilename**: (προαιρετικά) προτεινόμενο όνομα αρχείου
- **filefilter**: (προαιρετικά) κριτήριο (**string**) που καθορίζει τα αρχεία που μπορούμε να επιλέξουμε βάσει της επέκτασής τους
- **filterindex**: (προαιρετικά) δείκτης που καθορίζει προτεινόμενο φίλτρου
- **title**: (προαιρετικά) ο τίτλος του πλαισίου διαλόγου
- **buttontext**: (προαιρετικά) μόνο για την Macintosh έκδοση του Excel

Το αντικείμενο FileDialog

Αυτό το αντικείμενο μας δίνει τη δυνατότητα να εμφανίσουμε τέσσερεις τύπους πλαισίων διαλόγου. Χρησιμοποιώντας την αντίστοιχη ιδιότητα του αντικειμένο Application επιστρέφεται ένα αντικείμενο FileDialog, ο τύπος του οποίου καθορίζεται από την τιμή του μοναδικού ορίσματος **DialogType** της ιδιότητας, όπως παρακάτω:

Τιμή	Τύπος αντικειμένου
msoFileDialog OpenDialogBox	Επιτρέπει την επιλογή ενός ή περισσοτέρων αρχείων που θα ανοιχθούν, χρησιμοποιώντας τη μέθοδο **Execute**
msoFileDialog SaveAs	Επιτρέπει την επιλογή ενός ονόματος αρχείου για αποθήκευση, χρησιμοποιώντας τη μέθοδο **Execute**
msoFileDialog FilePicker	Επιτρέπει την επιλογή ενός ή περισσοτέρων αρχείων και επιστρέφει τις διαδρομές τους στη συλλογή **FileDialogSelectedItems**
msoFileDialog FolderPicker	Επιτρέπει την επιλογή μίας διαδρομής και επιστρέφει αυτή στη συλλογή **FileDialogSelectedItems**

Πίνακας 13 Τιμές ορίσματος DialogType της ιδιότητας FileDialog

Στο παρακάτω παράδειγμα επιλέγουμε ένα φάκελο και εμφανίζουμε τη διαδρομή του σε ένα μήνυμα. Η μέθοδος **show** εμφανίζει το πλαίσιο διαλόγου.

```
Sub GetAFolder ()

   With Application.FileDialog(msoFileDialogFolderPicker)

'  Δηλώνουμε την προκαθορισμένη διαδρομή
      .InitialFileName = Application.DefaultFilePath & "\"
      .Title = "Επιλέξτε φάκελο"
'  Εμφανίζουμε το παλίσιο διαλόγου
      .Show
'  Ελέγχουμε αν έχει επιλεχθεί ένας φάκελος
      If .SelectedItems.Count = 0 Then
         MsgBox "Δεν επιλέξατε φάκελο"
      Else
'  Εμφανίζουμε τη διαδρομή του επιλεγμένου φακέλου
         MsgBox "Επιλέξατε το φάκελο: " & .SelectedItems(1)
      End If
   End With

End Sub
```

Φόρμες χρήστη (User Forms)

Όταν θέλουμε να σχεδιάσουμε ένα πλαίσιο διαλόγου από την αρχή, που θα εκτελεί συγκεκριμένες λειτουργίες, χρησιμοποιούμε μία φόρμα χρήστη. Τα βασικά βήματα που ακολουθούμε είναι τα εξής:

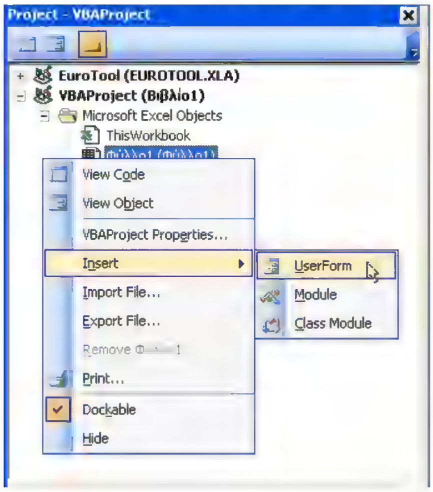
- Εισάγουμε μία φόρμα στο παράθυρο του Rproject Explorer

- Δημιουργούμε μία διαδικασία που θα εμφανίσει τη φόρμα, την οποία αποθηκεύουμε σε ένα module και όχι στην ίδια τη φόρμα
- Προσθέτουμε τα στοιχεία ελέγχου (controls) που θέλουμε στη φόρμα
- Καθορίζουμε όπου απαιτείται τις ιδιότητες των στοιχείων ελέγχου
- Δημιουργούμε διαδικασίες διαχείρισης συμβάντων για τα στοιχεία ελέγχου, τις οποίες αποθηκεύουμε στη φόρμα και εκτελούνται όταν ανιχνεύονται τα αντίστοιχα συμβάντα

Κάθε φόρμα είναι και ένα αντικείμενο **UserForm**.

Εισαγωγή μίας φόρμας

Για να εισάγουμε μία φόρμα επιλέγουμε το VBA project στον Project Explorer και στη συνέχεια πατάμε το δεξί κουμπί του ποντικιού και στο μενού που εμφανίζεται επιλέγουμε **Insert – UserForm**.

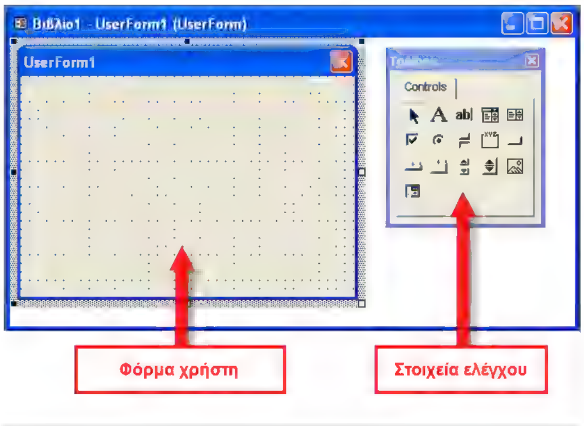


Εικόνα 60 Εισαγωγή μίας φόρμας

Η φόρμα που δημιουργήθηκε παίρνει αυτόματα ένα όνομα της μορφής UserFormN, το οποίο μπορούμε να τροποποιήσουμε στο παράθυρο ιδιοτήτων. Ένα βιβλίο εργασίας μπορεί να περιέχει πολλές φόρμες.

Στοιχεία ελέγχου

Για να προσθέσουμε στοιχεία ελέγχου χρησιμοποιούμε την **εργαλειοθήκη** (**toolbox**). Αν δεν είναι εμφανής επιλέγουμε από το μενού **View-Toolbox**. Επιλέγουμε το στοιχείο που θέλουμε και στη συνέχεια κάνουμε κλικ μέσα στο πλαίσιο διαλόγου για να προστεθεί το στοιχείο ελέγχου. Στο παράθυρο Ιδιότητες μπορούμε να τροποποιησούμε τις ιδιότητες, αν απαιτείται.

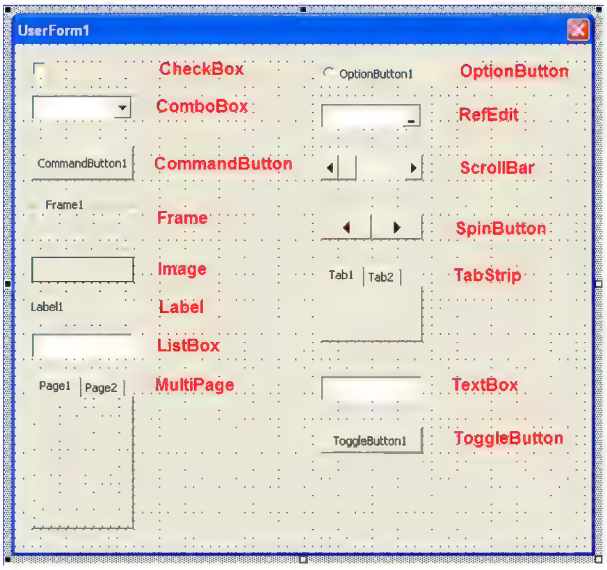


Εικόνα 61 Μία φόρμα και τα στοιχεία ελέγχου που μπορούμε να προσθέσουμε

Τα διαθέσιμα στοιχεία ελέγχου είναι τα παρακάτω:

Στοιχείο ελέγχου	Περιγραφή
CheckBox	Το στοιχείο αυτό έχει τιμή true όταν είναι επιλεγμένο και όταν δεν είναι έχει τιμή false
ComboBox	Πλαίσιο που περιέχει μία λίστα στοιχείων (items) και επιτρέπει την επιλογή ενός και μόνο στοιχείου
CommandButton	Χρησιμοποιείται συνήθως με το συμβάν click (που ανιχνεύεται όταν πατάμε με το ποντίκι) για να καλέσει μία διαδικασία
Frame	Περικλείει άλλα στοιχεία ελέγχου, συνήθως optionbuttons, τα οποία ομαδοποιεί λογικά
Image	Εμφανίζει μία εικόνα από αρχείο ή από επικόλληση
Label	Εμφανίζει μία ετικέτα
ListBox	Πλαίσιο που περιέχει μία λίστα στοιχείων και επιτρέπει την επιλογή ενός ή περισσότερων στοιχείων
Multipage	Δημιουργεί καρτέλες, που μπορούν να περιέχουν άλλα στοιχεία ελέγχου
OptionButton	Χρησιμοποιούνται συνήθως περισσότερα από, ομαδοποιημένα. Όταν επιλεγεί ένα από την ομάδα αποεπιλέγονται τα υπόλοιπα
RefEdit	Επιτρέπει την επιλογή μίας περιοχής από ένα φύλλο εργασίας
ScrollBar	Παρέχει τη δυνατότητα επιλογής τιμών με τη χρήση του ποντικιού
SpinButton	Παρέχει τη δυνατότητα επιλογής τιμών βηματικά με τη χρήση του ποντικιού
TabStrip	Παρόμοιο με το MultiPage, αλλά δεν πμορεί να περιέχει άλλα στοιχεία ελέγχου
TextBox	Πλαίσιο στο οποίο εισάγουμε κείμενο
ToggleButton	Έχει τιμή true όταν είναι πατημένο και false όταν δεν είναι

Πίνακας 14 Τα διαθέσιμα στοιχεία ελέγχου



Εικόνα 62 Τα διαθέσιμα στοιχεία ελέγχου τοποθετημένα σε μία φόρμα

Εμφάνιση και κλείσιμο μίας φόρμας

Προκειμένου να εμφανίσουμε μία φόρμα, δημιουργούμε μία διαδικασία που αποθηκεύουμε σε ένα module και εκτελούμε τη μέθοδο **Show** του αντικειμένου **UserForm**. Αν για παράδειγμα η φόρμα μας ονομάζεται **UserForm1** τότε η διαδικασία που θα την εμφανίσει θα είναι όπως παρακάτω:

```
Sub DisplayForm()
   UserForm1.Show
End Sub
```

Αν το όνομα μίας φόρμας είναι μεταβλητή τύπου **string**, πρέπει πρώτα να χρησιμοποιήσουμε τη μέθοδο **Add** για να την προσθέσουμε στη συλλογή **UserForms** και στη συνέχεια τη μέθοδο **Show** για να την εμφανίσουμε:

```
Form = "UserForm1"
UserForms.Add(UserForm1).Show
```

Η φόρμα παραμένει ενεργή μέχρι να την κλείσουμε. Συνήθως αυτό γίνεται με ένα CommandButton που καλεί μία διαδικασία που χρησιμοποιεί τη μέθοδο **hide** ή τη μέθοδο **unload**.

```
Sub CloseForm()
   Unload UserForm1
End Sub
```

Μπορούμε όμως και να γράψουμε την παρακάτω εντολή, στο κώδικα της ίδιας της φόρμας:

```
Unload Me
```

Μέχρι να κλείσει η φόρμα δεν είναι δυνατόν να εργαστούμε στο φύλλο εργασίας. Αν επιθυμούμε να εργαζόμαστε στο φύλλο και η φόρμα να παραμένει ορατή, θα πρέπει να δηλώσουμε ότι η φόρμα είναι modeless, ως εξής:

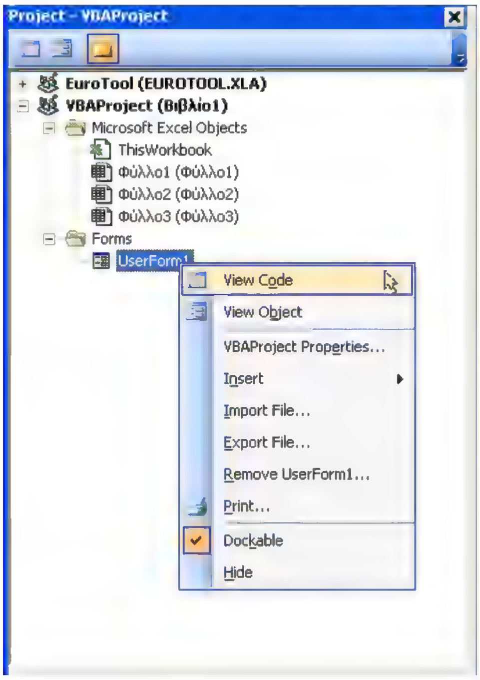
UserForm1.Show 0

Όταν κλείνουμε μία φόρμα με τη μέθοδο **Unload** τα στοιχεία ελέγχου αρχικοποιούνται και χάνουν τυχόν αλλαγές των τιμών τους.

Οι διαδικασίες διαχείρισης συμβάντων

Από τη στιγμή που θα εμφανιστεί μία φόρμα, ο χρήστης αλληλεπιδρά με αυτή, μέσω των στοιχείων ελέγχου που περιέχει. Για παράδειγμα κάνει κλικ σε ένα κουμπί **CommandButton**, ή επιλέγει μία τιμή σε ένα **ListBox** κλπ. Στην πραγματικότητα προκαλεί κάποια συμβάντα. Τα συμβάντα ανιχνεύονται από το Excel και θα πρέπει ο κώδικας μας να ανταποκρίνεται ανάλογα, χρησιμοποιώντας τις προκαθορισμένες διαδικασίες διαχείρισης συμβάντων του Excel για τα στοιχεία ελέγχου. Όταν για παράδειγμα κάνουμε κλικ σε ένα **CommandButton1** προκαλείται το συμβάν **Click** και η αντίστοιχη διαδικασία διαχείρισης συμβάντων είναι η **CommandButton1_Click()**.

Ο κώδικας των διαδικασιών αυτών βρίσκεται εντός του UserForm. Για να δούμε τον κώδικα πρέπει να κάνουμε δεξί κλικ στο UserForm, στο παράθυρο του Project Explorer και να επιλέξουμε **View Code**.



Εικόνα 63 Πρόσβαση στον κώδικα μίας φόρμας

Ολοκληρωμένο παράδειγμα δημιουργίας και διαχείρισης φόρμας

Στο παράδειγμα θα δημιουργήσουμε μία φόρμα, θα προσθέσουμε τα απαιραίτητα στοιχεία ελέγχου και θα γράψουμε τον κώδικα που απαιτείται, προκειμένου να είναι λειτουργική η φόρμα μας.

Πως δημιουργούμε τη φόρμα:

1. Στο παράθυρο του Project Exporer κάνουμε δεξί κλικ και επιλέγουμε **Insert-UserForm**.
2. Στο παράθυρο ιδιοτήτων αλλάζουμε την τιμή της ιδιότητα Caption σε Προσωπικά Στοιχεία.
3. Προσθέτουμε ένα στοιχείο MultiPage και τροποποιούμε τις ιδιότητες της Page1 ως εξής:
   - Caption: Όνομαματεπώνυμο και Φύλλο
4. Τροποποιούμε τις ιδιότητες της Page2 ως εξής:
   - Caption: Λοιπά Στοιχεία
5. Προσθέτουμε ένα στοιχείο Label και τροποποιούμε τις ιδιότητες του ως εξής:
   - Caption: Όνομαματεπώνυμο:
   - TabIndex: 0
6. Προσθέτουμε ένα TextBox και τροποποιούμε τις ιδιότητες του ως εξής:
   - Name: TextBoxFullName
   - TabIndex: 1
7. Προσθέτουμε ένα Farme και τροποποιούμε τις ιδιότητες του ως εξής:
   - Caption: Φύλλο
   - TabIndex: 2
8. Προσθέτουμε ένα OptionButton μέσα στο Frame και τροποποιούμε τις ιδιότητες του ως εξής:
   - Caption: Άνδρας
   - Name: OptionButtonMale
   - TabIndex: 0
9. Προσθέτουμε ένα δεύτερο OptionButton μέσα στο Frame και τροποποιούμε τις ιδιότητες του ως εξής:
   - Caption: Γυναίκα
   - Name: OptionButtonFemale
   - TabIndex: 1
10. Προσθέτουμε ένα Label στην Page2 και τροποποιούμε τις ιδιότητες του ως εξής:
    - Caption: Περιοχή Κατοικίας:
11. Προσθέτουμε ένα RefEdit στην Page2 και τροποποιούμε τις ιδιότητες του ως εξής:
    - Name: RefEditArea
    - TabIndex: 0
12. Προσθέτουμε ένα Label στην Page2 και τροποποιούμε τις ιδιότητες του ως εξής:
    - Caption: Επάγγελμα:
13. Προσθέτουμε ένα ComboBox στην Page2 και τροποποιούμε τις ιδιότητες του ως εξής:
    - Name: RefEditProf
    - TabIndex: 1
14. Προσθέτουμε ένα Label στην Page2 και τροποποιούμε τις ιδιότητες του ως εξής:
    - Caption: Hobby:
15. Προσθέτουμε ένα ListBox στην Page2 και τροποποιούμε τις ιδιότητες του ως εξής:
    - Name: RefEditProf
    - TabIndex: 2

16. Προσθέτουμε ένα Label στην Page2 και τροποποιούμε τις ιδιότητες του ως εξής:
    - Caption: Ηλικία:
17. Προσθέτουμε ένα SpinButton στην Page2 και τροποποιούμε τις ιδιότητες του ως εξής:
    - Name: SpinButtonAge
    - TabIndex: 3:
    - Min: 1
    - Max: 120
18. Προσθέτουμε ένα TextBox στην Page2 και τροποποιούμε τις ιδιότητες του ως εξής:
    - Name: TextBoxAge
    - Enabled: False
    - Text: ?
19. Προσθέτουμε ένα CommandButton (εκτός MultiPage) και τροποποιούμε τις ιδιότητες του ως εξής:
    - Caption: OK
    - Default: True
    - Name: OKButton
    - TabIndex: 1
20. Προσθέτουμε ένα δεύτερο CommandButton (εκτός MultiPage) και τροποποιούμε τις ιδιότητες του ως εξής:
    - Caption: Ακύρωση
    - Cancel: True
    - Name: CancelButton
    - TabIndex: 0

Η φόρμα μας πρέπει να είναι όπως στην απόμενη εικόνα.

Πως προσθέτουμε τις διαδικασίες διαχείρισης συμβάντων:

1. Κάνουμε διπλό κλικ στο CommandButton Ακύρωση και εμφανίζεται το παράθυρο κώδικα για τη φόρμα με μία κενή διαδικασία **CommandButtonCancel_Click()** (είναι η διαδικασία διαχείρισης του συμβαντος **Click** του CommandButtonCancel), την οποία τροποιούμε ως εξής:

```
Private Sub CommandButtonCancel_Click()
    Unload Me
End Sub
```

   Η διαδικασία εκτελείται όταν κάνουμε κλικ στο κουμπί Ακύρωση και κλείνει τη φόρμα.

2. Κάνουμε διπλό κλικ στο CommandButtonOK και εμφανίζεται το παράθυρο κώδικα για τη φόρμα με μία κενή διαδικασία **CommandButtonOK_Click()** (είναι η διαδικασία διαχείρισης του συμβαντος **Click** του CommandButtonCancel). Τη διαδικασία αυτή την τροποποιούμε ως εξής:

```
Private Sub OKButton_Click()
```

```
    Dim AreaRange As Range

    Sheets("Sheet1").Activate

    Cells(1, 2) = TextBoxFulName.Text
    If OptionButtonMale Then Cells(2, 2) = "Άνδρας"
    If OptionButtonFemale Then Cells(2, 2) = "Γυναίκα"
    Set AreaRange = Range(RefEditArea.Text).Range("A1")
    Cells(3, 2) = AreaRange.Value
    Cells(4, 2) = Prof(ComboBoxProf.ListIndex)
    Cells(5, 2) = Hobby(ListBoxHobby.ListIndex)
    Cells(6, 2) = TextBoxAge.Text

    Unload Me

End Sub
```

Η διαδικασία εκτελείται όταν κάνουμε κλικ στο κουμπί OK και καταχωρεί τις τιμές που επιλέξαμε στο φύλλο εργασίας.

Η ιδιότητα **ListIndex** των αντικειμένων **ComboBox** και **ListBox**, περιέχει τη σχετική θέση της τιμής που επιλέξαμε στα δύο αυτά στοιχεία ελέγχου. Οι τιμές που επιστρέφει η **ListIndex** κυμαίνονται μεταξύ -1 και του πλήθους των τιμών που περιέχουν τα αντικείμενα μείον 1. Χρησιμοποιούμε την ιδιότητα **ListIndex** ως δείκτη στους πίνακες **Hobby** και **Prof**, προκειμένου να εισάγουμε στα κελιά του φύλλου εργασίας τις τιμές που επιλέξαμε, οι οποίες είναι αποθηκευμένες σε αυτούς τους πίνακες.

3. Κάνουμε διπλό κλικ στο SpinButtonAge και εμφανίζεται το παράθυρο κώδικα για τη φόρμα με μία κενή διαδικασία **SpinButtonAge_Change()** (είναι η διαδικασία διαχείρισης του συμβαντος **Change** του SpinButtonAge), την οποία τροποποιούμε ως εξής:

```
Private Sub SpinButtonAge_Change()

    TextBoxAge.Text = SpinButtonAge.Value

End Sub
```

Η διαδικασία αυτή εκτελείται όταν κάνουμε κλικ σ' ένα από τα δύο κουμπιά του SpinButtonAge και αποδίδει την τιμή αυτού στο TextBoxAge

4. Στο παράθυρο κώδικα της φόρμας επιλέγουμε στο Object List Box το αντικείμενο UserForm και στο Procedure List Box τη διαδικασία Initialize. Εμφανίζεται η κενή διαδικασία UserForm_Initialize() (είναι η διαδικασία διαχείρισης του συμβαντος **Initialize** του UserForm), την οποία τροποποιούμε ως εξής:

```
Private Sub UserForm_Initialize()

    Hobby(0) = "Αθλητισμός"
    Hobby(1) = "Μουσική"
    Hobby(2) = "Κινηματογράφος"

    Prof(0) = "Ιδιωτικός Υπαλληλος"
    Prof(1) = "Δημόσιος Υπάλληλος"
    Prof(2) = "Ελεύθερος Επαγγελματίας"
```

```
        Prof(3) = "Συνταξιούχος"

        ' Προσθέτουμε τις τιμές που αποδώσαμε στον πίνακα Hobby
        ' στο ListBoxHobby
        With ListBoxHobby
            .AddItem Hobby(0)
            .AddItem Hobby(1)
            .AddItem Hobby(2)
        End With
        ' Προσθέτουμε τις τιμές που αποδώσαμε στον πίνακα Prof
        ' στο ComboBoxProf
        With ComboBoxProf
            .AddItem Prof(0)
            .AddItem Prof(1)
            .AddItem Prof(2)
            .AddItem Prof(3)
        End With

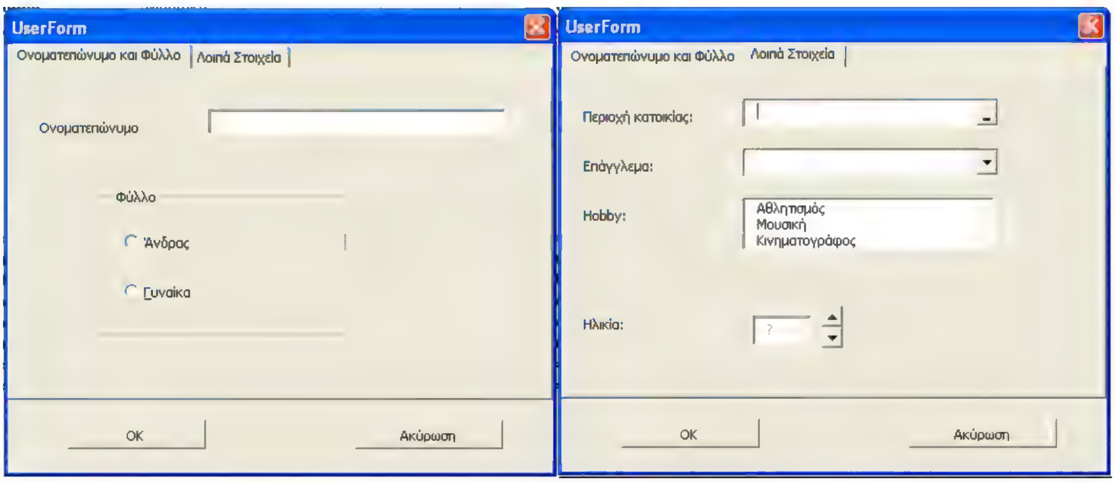
    End Sub
```

Η διαδικασία αυτή εκτελείται πριν εμφανιστεί η φόρμα και αρχικοποιεί τις τιμές του array **Hobby(2)** και του array **Prof(3)**, τα οποία περιέχουν τις τιμές που θα εμφανίστουν στα **ListBoxHobby** και **ComboBoxProf** αντίστοιχα. Τα δύο αυτά arrays έχουν δηλωθεί στην αρχή του module που περιέχει τον κώδικα της φόρμας και είναι ορατά καθόλη τη διάρκεια που η φόρμα είναι ενεργή. Τα στοιχεία ελέγχου ComboBox και ListBox επιστρέφουν ένα δείκτη (**ListIndex**) και όχι την τιμή που εμφανίζουν στη φόρμα.

5. Οι δύο πίνακες **Hobby(2)** και **Prof(3)**, πρέπει να δηλωθούν στην αρχή του κώδικα που περιέχεται στη φόρμα, ώστε να είναι ορατές από όλες τις διαδικασίες που περιέχει η φόρμα:

```
Dim Hobby(2)
Dim Prof(3)
```

Οι δύο καρτέλες της φόρμας εμφανίζονται στην επόμενη εικόνα.

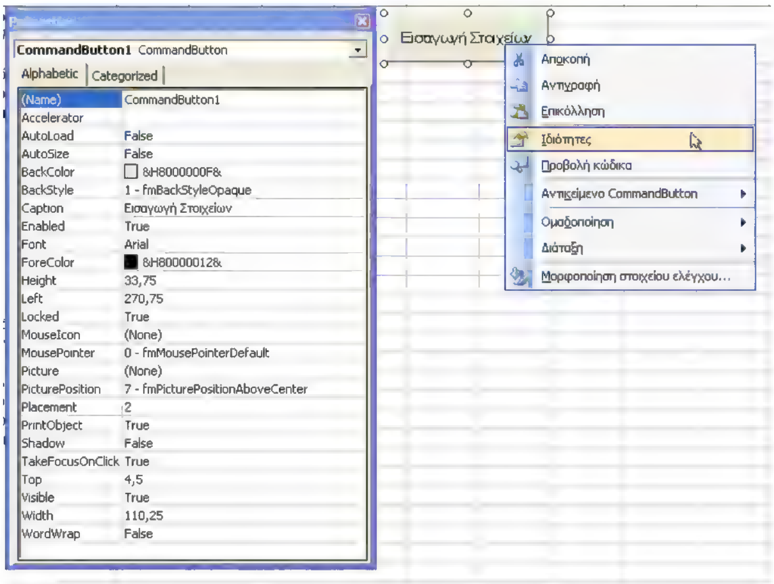


Εικόνα 64 Η φόρμα που δημιουργήσαμε

Για να εμφανίσουμε τη φόρμα εισάγουμε στο φύλλο εργασίας ένα CommandButton, (από την εργαλειοθήκη στοιχείων ελέγχου). Σε κατάσταση σχεδίασης (αν δεν είμαστε σε κατάσταση σχεδίασης κάνομυ κλικ στο πρώτο κουμπί της εργαλιοθήκης) κάνουμε δεξί κλικ στο CommandButton, επιλέγουμε **Ιδιότητες** και τροποποιούμε την ιδιότητα Caption σε Εισαγωγή Στοιχείων στο παράθυρο ιδιοτήτων. Στη συνέχεια κάνουμε πάλι δεξί κλικ επιλέγουμε Προβολή Κώδικα και εμφανίζεται το παράθυρο κώδικα για το CommandButton με μία κενή διαδικασία **CommandButton_Click()** (είναι η διαδικασία διαχείρισης του συμβαντος **Click** του CommandButton), την οποία τροποποιούμε ως εξής:

```
Private Sub CommandButton1_Click()
    UserForm.Show
End Sub
```

Η διαδικασία αυτή εκτελείται όταν κάνουμε κλικ στο CommandButton και εμφανίζει τη φόρμα που δημιουργήσαμε.

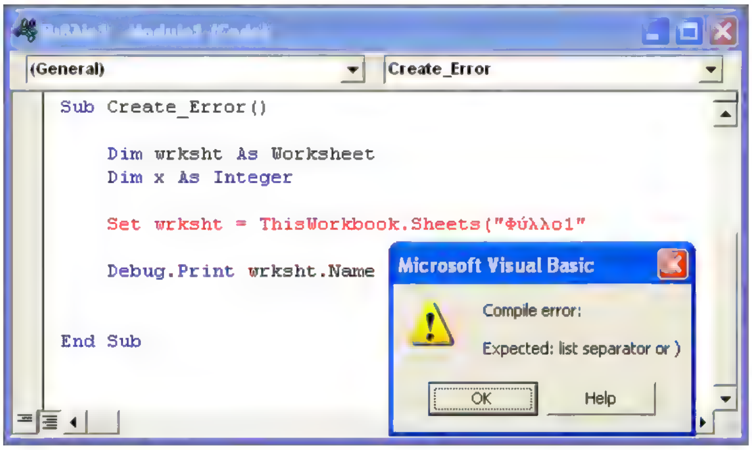


Εικόνα 65 Το CommandButton που καλεί τη φόρμα

Διαχείριση σφαλμάτων

Είδη σφαλμάτων

Μία διαδικασία μπορεί να περιέχει **σφάλματα σύνταξης** (**syntax errors**) τα οποία θα ανιχνευτούν από τον επεξεργαστή της VBA καθώς πληκτρολογούμε την έκφραση. Ο επεξεργαστής της VBA επισημαίνει αμέσως το σφάλμα και εμφανίζεται σχετικό μήνυμα που μας πληροφορεί ότι απαιτείται να διορθώσουμε το σφάλμα.



Εικόνα 66 Σφάλμα σύνταξης και σχετικό μήνυμα

Ο κώδικας VBA πρέπει να μεταφραστεί (**compile**) πριν να εκτελεστεί. Η μεταγλώττιση γίνεται αυτόματα μόλις προσπαθήσουμε να εκτελέσουμε τη διαδικασία που περιέχει τον κώδικα. Ένα είδος σφαλμάτων εντοπίζεται κατά τη μεταγλώττιση, και ονομάζονται σφάλματα μεταγλώττισης (**compile errors**). Όταν υπάρχουν τέτοια σφάλματα δεν είναι δυνατή η εκτέλεση του κώδικα μίας διαδικασίας. Μόνο μετά την διόρθωση είναι δυνατή η εκτέλεση.

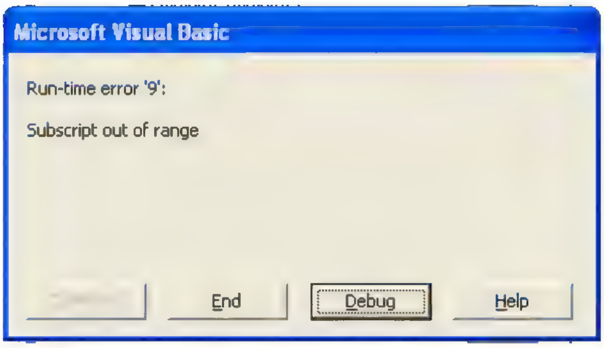
Είναι δυνατόν ο κώδικας να μην περιέχει σφάλματα σύνταξης ή μεταγλώττισης, αλλά να παρουσιάζονται σφάλματα κατά την εκτέλεση του. Τα σφάλματα αυτά ονομάζονται σφάλματα κατά την εκτέλεση (**run time errors**). Σε αυτή την περίπτωση σταματάει η εκτέλεση και εμφανίζεται ένα μήνυμα με τον κωδικό του σφάλματος και μία περιγραφή.

Στο επόμενο παράδειγμα, προσπαθούμε να αλλάξουμε το όνομα του τέταρτου φύλλου, του ενεργού βιβλίου εργασίας. Επειδή όμως το βιβλίο μας έχει μόνο τρία φύλλα εμφανίζεται ένα μήνυμα σφάλματος.

```
Sub Create_Error()

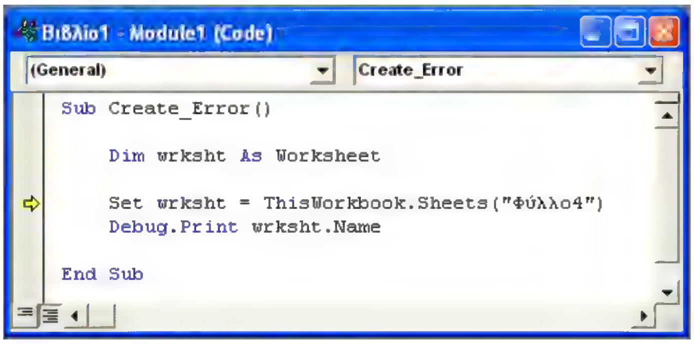
  Dim wrksht as Worksheet
  Set wrksht = ThiwWorkbook.Sheets("Φύλλο4")
  wrksht.Name = "Συνολικό άθροισμα"

End Sub
```



Εικόνα 67 Μήνυμα σφάλματος

Μπορούμε να επιλέξουμε να τερματίσουμε την εκτέλεση του κώδικα, ή να διορθώσουμε το σφάλμα δηλαδή να κάνουμε **αποσφαλμάτωση** (**debug**). Στη δεύτερη περίπτωση εμφανίζεται το παράθυρο κώδικα και η εντολή όπου σταμάτησε η εκτέλεση επισημασμένη.



Εικόνα 68 Η εντολή που περιέχει το σφάλμα

Για να συνεχίσουμε πρέπει πρώτα να διορθώσουμε το σφάλμα και στη συνέχεια να συνεχίσουμε την εκτέλεση (πατώντας **F5** ή **F8**, ή από την επιλογή μενού **Debug – Step Into** ή **Run – Continue**).

Υπάρχουν περιπτώσεις όμως όπου ακόμη και να διορθώσουμε το σφάλμα δεν είναι δυνατή η συνέχιση της εκτέλεσης. Τότε πρέπει να σταματήσουμε την εκτέλεση από την επιλογή μενού **Run – Reset**.

Μία ιδιαίτερη κατηγορία σφαλμάτων είναι τα σφάλματα λογικής, τα οποία δεν εμποδίζουν την εκτέλεση του κώδικα. Προκαλούν όμως μη επιθυμητά αποτελέσματα.

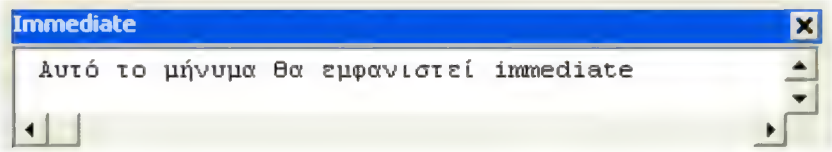
Εργαλεία και μέθοδοι για την αποσφαλμάτωση

Ο επεξεργαστής της VBA μας παρέχει μερικά ιδιαίτερα χρήσιμα εργαλεία.

Το παράθυρο Immediate

Μπορούμε να χρησιμοποιούμε το παράθυρο Immediate για να εκτελούμε άμεσα εκφράσεις της VBA. Αρκεί να γράψουμε την έκφραση και να πατήσουμε **Enter**. ή να εμφανίζουμε πληροφορίες με τη βοήθεια της εντολής **Debug.Print**.

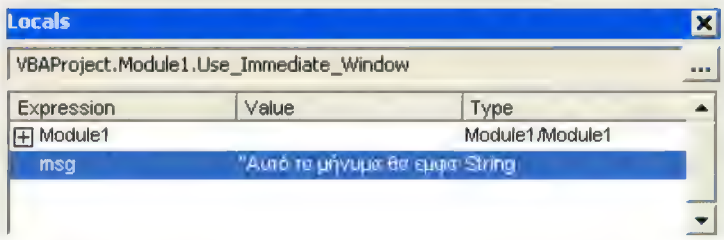
```
Sub Use_Immediate_Window()
    Dim msg as String
    msg = "Αυτό το μήνυμα θα εμφανιστεί στο παράθυρο immediate"
End Sub
```



Εικόνα 69 Εμφάνιση πληροφοριών στο παράθυρο immediate

Το παράθυρο Locals

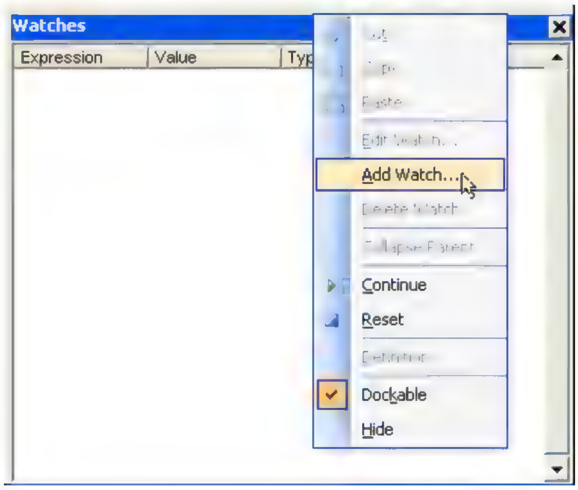
Εμφανίζονται οι τιμές μεταβλητών κατά τη βηματική εκτέλεση του κώδικα. Μπορούμε επίσης να μεταβάλλουμε τις τιμές τους.



Εικόνα 70 Το παράθυρο locals

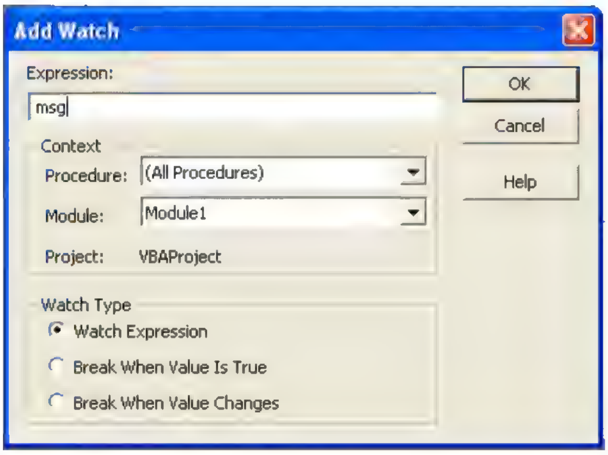
Το παράθυρο watches

Στο παράθυρο αυτό μπορούμε να επιλέξουμε συγκεκριμένες μεταβλητές, την τιμή των οποίων θέλουμε να παρακολουθούμε συνεχώς. Κάνουμε δεξί κλικ και επιλέγουμε **Add Watch**.



Εικόνα 71 Προσθήκη μεταβλητής στο παράθυρο Watches

Γράφουμε τη μεταβλητή που θέλουμε, επιλέγουμε αν θέλουμε να παρακολουθούμε την τιμή του (**Watch Type: Watch Expression**), ή αν θέλουμε να σταματήσει η εκτέλεση όταν η τιμή της είναι True ή όταν μεταβληθεί (**Watch Type: Break When Value Is True** ή **Break When Value Changes** αντίστοιχα).

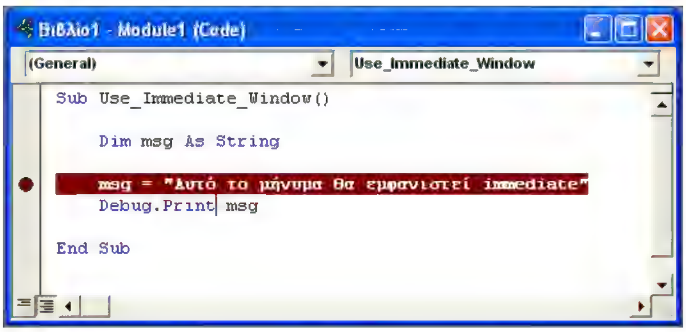


Εικόνα 72 Επιλογή παραμέτρων για την μεταβλητή υπό παρακολούθηση

Βηματική εκτέλεση κώδικα και breakpoints

Πολλές φορές δεν είναι εύκολο να καταλάβουμε που βρίσκεται το σφάλμα. Μία μέθοδος που μπορεί να μας βοηθήσει στην περίπτωση όπου τα σφάλματα είναι σχεδιαστικά ή λογικά, είναι η **βηματική** εκτέλεση του κώδικα εντολή προς εντολή (με **F8** ή από την επιλογή μενού **Debug – Step Into**). Με τη βηματική εκτέλεση έχουμε τη δυνατότητα να παρακολουθούμε τιμές των μεταβλητών και να εντοπίζουμε λογικά – σχεδιαστικά σφάλματα στον κώδικα.

Αν ο κώδικας έχει πολλές γραμμές εντολών ή έχουμε πολλά διαδικασίες ή modules σε ένα project, αντί να εκτελέσουμε βηματικά τον κώδικα μπορούμε να χρησιμοποιήσουμε **breakpoints**. Εισάγουμε breakpoint σε μία εντολή από την επιλογή μενού **Debug – Toggle Breakpoint** ή πατώντας **F9**.



Εικόνα 73 Breakpoint

Εκτελούμε τον κώδικα πατώντας **F5** ή από τη επιλογή μενού **Run – Run Sub/UserForm**. Μόλις η εκτέλεση φτάσει στην εντολή που έχουμε εισάγει το breakpoint σταματάει η εκτέλεση και μπορούμε να δούμε τις μεταβλητές.

Διαχείριση σφαλμάτων

Ακόμη και στην περίπτωση όπου έχουμε ελέγξει σχολαστικά τον κώδικα που έχουμε γράψει, είναι πιθανό να παρουσιαστούν περιπτώσεις σφαλμάτων κατά την εκτέλεση του κώδικα σε πραγματικές συνθήκες. Είναι γενικά καλή πρακτική να χρησιμοποιούμε τεχνικές **διαχείρισης σφαλμάτων**, προκειμένου να αντιμετωπιστούν ανάλογες περιπτώσεις, χωρίς να απαιτείται να σταματάει η εκτέλεση του κώδικα.

Μπορούμε να εξασφαλίσουμε ότι στην περίπτωση όπου εμφανιστεί ένα σφάλμα, δεν θα σταματήσει η εκτέλεση του κώδικα μία διαδικασίας, αν εισάγουμε αμέσως μετά τη δήλωση της την έκφραση **On Error Resume Next**. Όταν κατά την εκτέλεση μίας έκφρασης VBA, ανιχνευτεί ένα σφάλμα ο επεξεργαστής της VBA την αγνοεί και συνεχίζει εκτελώντας την επόμενη έκφραση.

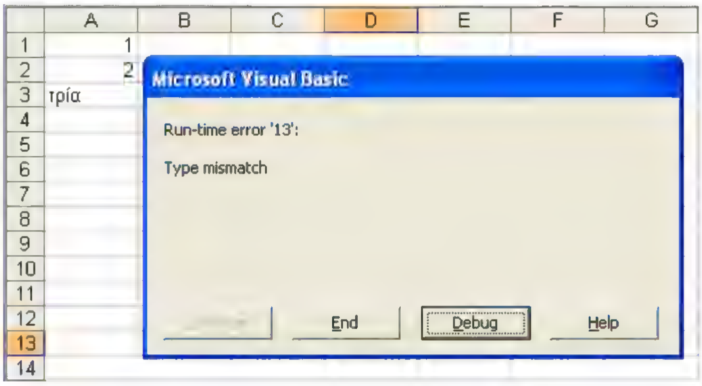
Στο επόμενο παράδειγμα, υπολογίζουμε το άθροισμα των κελιών A1, B1, C1 και εισάγουμε το αποτέλεσμα στο κελί D1. Ο κώδικας δεν περιέχει κανένα σφάλμα, αν όμως σε κάποιο από τα κελιά δεν υπάρχει αριθμός αλλά κείμενο θα εμφανιστεί μήνυμα σφάλματος κατά την εκτέλεση.

```
Sub Ignore_Error()

    Dim Total As Long
    Dim Num1 As Long
    Dim Num2 As Long
    Dim Num3 As Long

    Num1 = Cells(1, 1)
    Num2 = Cells(2, 1)
    Num3 = Cells(3, 1)
    Total = Num1 + Num2 + Num3
    Cells(4, 1) = Total

End Sub
```



Εικόνα 74 Μήνυμα σφάλματος που εμφανίζεται λόγω του περιεχομένου του κελιού C1

Αν όμως εισάγουμε την έκφραση **On Error Resume Next**, η εκτέλεση θα συνεχιστεί αγνοώντας το κελί C1 και θα εισάγει στο κελί D4 το άθροισμα των δύο πρώτων κελιών.

Η VBA μας δίνει τη δυνατότητα να εκτελέσουμε συγκεκριμένο κώδικα όταν ανιχνευθεί ένα σφάλμα. Με τον κώδικα αυτό συνήθως εμφανίζουμε μηνύματα που πληροφορούν το χρήστη για τις συνθήκες που δημιούργησαν το σφάλμα και του παρέχουν οδηγίες για να το αντιμετωπίσει. Η έκφραση που χρησιμοποιούμε έχει την παρακάτω σύνταξη:

```
Sub ...()
    On Error GoTo Error_Handler
        ... κώδικας διαδικασίας ...
    Exit Sub
    Error_Handler:
        ... κώδικας διαχείρισης σφάλματος ....
    Resume Next
End Sub
```

κώδικας διαχείρισης σφάλματος

όπου:

On Error GoTo: έκφραση που μεταφέρει την εκτέλεση σε άλλο σημείο του κώδικα

Error_Handler: ετικέτα (**label**) που χρησιμοποιείται ως όνομα για τον κώδικα που διαχειρίζεται το σφάλμα, για να δηλώνουμε στη VBA σε ποιο σημείο του κώδικα της διαδικασίας θέλουμε να μεταβεί η εκτέλεση μόλις προκύψει ένα σφάλμα.

Exit Sub: έκφραση που διακόπτει την εκτέλεση του κώδικα μίας διαδικασίας και μας «βγάζει» από αυτή. Στην περίπτωση όπου η συγκεκριμένη διαδικασία είχε κληθεί από άλλη, η εκτέλεση συνεχίζεται από το σημείο της δεύτερης που είχε πραγματοποιηθεί η κλήση

Resume Next: έκφραση που επιστρέφει την εκτέλεση στην επόμενη εντολή από αυτή που προκάλεσε το σφάλμα

Το προηγούμενο παράδειγμα μπορεί να τροποποιηθεί όπως παρακάτω:

```
Sub Handle_Error()

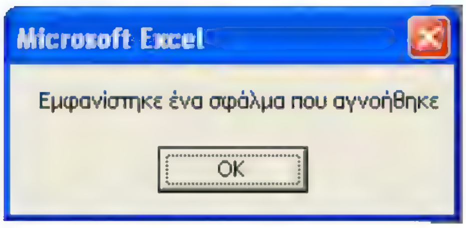
      On Error GoTo Error_Handler

  Dim Total As Long
  Dim Num1 As Long
  Dim Num2 As Long
  Dim Num3 As Long

  Num1 = Cells(1, 1)
  Num2 = Cells(2, 1)
  Num3 = Cells(3, 1)
  Total = Num1 + Num2 + Num3
  Cells(4, 1) = Total

  Error_Handler:
    MsgBox      "Εμφανίστηκε ένα σφάλμα που αγνοήθηκε"
  Resume Next

End Sub
```



Εικόνα 75 Το μήνυμα που εμφανίζεται κατά την διαχείριση του σφάλματος

Σύνδεση με βάσεις δεδομένων

Πρόσβαση σε βάσεις δεδομένων με τη βοήθεια της βιβλιοθήκης ADO

Η βιβλιοθήκη ADO (ActiveX Data Objects) παρέχει τη δυνατότητα σύνδεσης με βάσεις δεδομένων. Μέσω αυτής μπορούμε:

- να συνδεθούμε με οποιαδήποτε βάση δεδομένων που λειτουργεί σε περιβάλλον Windows
- να ανακτήσουμε δεδομένα από τη βάση (μέσω ερωτήματος)
- να εισάγουμε, τροποποιήσουμε ή να διαγράψουμε δεδομένα

Προϋπόθεση για τη χρήση των συναρτήσεων που περιέχει η βιβλιοθήκη στον κώδικα μας, είναι να την ενεργοποιήσουμε στο μενού **Tools – References**. Η βιβλιοθήκη που πρέπει να επιλέξουμε είναι η **Microsoft ActiveX Data Objects 2.8 Library** (ανάλογα με την έκδοση του Excel μπορεί να αλλάζει η έκδοση της).

Στο επόμενο παράδειγμα συνδεόμαστε με μία βάση δεδομένων Access για να ανακτήσουμε δεδομένα με τη βοήθεια της γλώσσας SQL (Structured Query Language).

```
Sub DBConnection()

  Dim conn As New Connection
  Dim recst As New Recordset
  Dim ws As Worksheet
  Dim query As String
  Dim i As Integer

  Set ws = ThisWorkbook.Worksheets("Sheet1")
  conn.Open "Provider=microsoft.jet.oledb.4.0;" & _
    "Data Source=C:\Program Files\Microsoft " &_
     Office\OFFICE11\SAMPLES\Northwind.mdb "
  query = "SELECT LastName, FirstName " & _
     "FROM employees ORDER BY LastName, FirstName"
  recst.Open query, conn

  n=0
  While Not recst.EOF
    n = n + 1
    ws.[A1].Cells(n) = recst!LastName & ", " & recst!FirstName
    recst.MoveNext
  Wend
 recst.Close
 conn.Close

End Sub
```

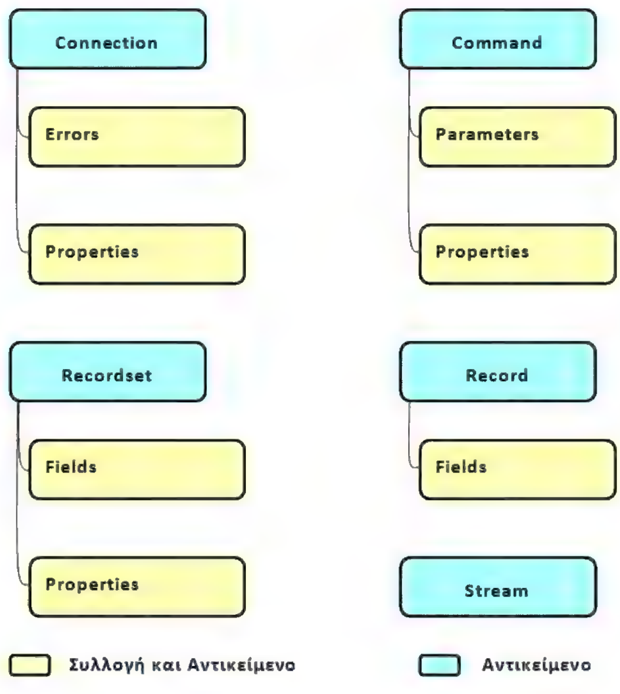
Το ADO Object Model

Η βιβλιοθήκη ADO περιέχει τα παρακάτω βασικά αντικείμενα:

Αντικείμενο	
Connection	Δημιουργεί τη σύνδεση με τη βάση δεδομένων
Command	Εντολή που περιέχει το ερώτημα (query)
Parammeters	Παράμετροι για το ερώτημα
Error	Μήνυμα σφάλματος
Recordset	Σύνολο εγγραφών δεδομένων (data records), όπως πίνακες, αποτελέσματα ερωτήματος κλπ
Fields	Πεδία των εγγραφών δεδομένων

Πίνακας 15 Αντικείμενα της βιβλιοθήκης ADO

Τα βασικά αντικείμενα του ADO Object Model που χρησιμοποιούμε είναι τα **Connection**, **Command** και **Resordset**. Προκειμένου να τα χρησιμοποιήσουμε αποδοτικά, είναι σημαντικό να γίνουν κατανοητές οι σχέσεις μεταξύ τους, που παρουσιάζονται στο παρακάτω σχήμα:



Εικόνα 76 ADO Object Model

Σε γενικές γραμμές τα βήματα που ακολουθούμε για να ανακτήσουμε δεδομένα από μία βάση είναι τα παρακάτω:

1. Δημιουργούμε ένα αντικείμενο **Connection** για να συνδεθούμε στη βάση δεδομένων που επιθυμούμε
2. Ανοίγουμε το αντικείμενο **Connection**
3. Δημιουργούμε ένα αντικείμενο **Command** που περιέχει μία ή περισσότερες εντολές που θα εκτελεστούν στη βάση δεδομένων
4. Αν απαιτούνται παράμετροι στο αντικείμενο **Command**, προσθέτουμε αντίστοιχα αντικείμενα **Parameter**
5. Δημιουργούμε ένα νέο αντικείμενο **Resordset**, με όλες τις αναγκαίες παραμέτρους
6. Περνάμε στην ιδιότητα **ActiveConnection** του **Recordset** το αντικείμενο **Connection** που δημιουργήσαμε προηγουμένως
7. Περνάμε στην ιδιότητα **ActiveCommand** του **Recordset** το αντικείμενο **Command** που δημιουργήσαμε προηγουμένως
8. Ανοίγουμε το αντικείμενο **Recordset**
9. Ανακτούμε τις εγγραφές που περιέχει το αντικείμενο **Recordset**
10. Κλείνουμε το αντικείμενο **Recordset**
11. Κλείνουμε το αντικείμενο **Connection**

Το αντικείμενο Connection

Δημιουργούμε τη σύνδεση με μία βάση δεδομένων, χρησιμοποιώντας τη μέθοδο **Open** του αντικειμένου **Connection**. Το αντικείμενο αυτό παρέχει και επίσης περιγράφει τη σύνδεση μεταξύ του Excel και της πηγής δεδομένων. Μία μεταβλητή μορφής **string** περνάει τις απαραίτητες παραμέτρους για τη σύνδεση. Με τη μέθοδο μπορούμε να συνδεθούμε με βάσεις δεδομένων Access, SQL Server, Oracle, ODBC και άλλες. Στην περίπτωση της Access η ορθή παράμετρος είναι:

Provider=microsoft.jet.oledb.4.0;Data Source=C:\path\name.mdb

όπου:

- **Provider=microsoft.jet.oledb.4.0**: δηλώνει ότι η βάση δεδομένων είναι Access
- **Data Source=C:\path\name.mdb**: δηλώνει τη διαδρομή και το αρχείο της βάσης δεδομένων Access

Οι κυριότερες μέθοδοι και οι ιδιότητες του αντικειμένου **Connection** παρουσιάζονται στους επόμενους πίνακες:

Ιδιότητα	
ConnectionString	Καθορίζει το τρόπο που θα γίνει η σύνδεση με τη βάση δεδομένων
CursorLocation	Καθορίζει αν ο τρέχον δείκτης εγγραφής (current record pointer) θα βρίσκεται στο Excel (adUseClient) ή στη βάση (adUseServer). Συνήθως η δεύτερη περίπτωση είναι προτιμότερη
Errors	Αναφέρεται σε συλλογή αντικειμένων Error που περιέχουν τα σφάλματα που παρουσιάστηκαν κατά τη σύνδεση
Mode	Καθορίζει πως θα γίνεται η πρόσβαση στη βάση δεδομένων. Οι τιμές μπορεί να είναι: • adModeRead • adModeWrite • adModeReadWrite • adModeShareDenyRead • adModeShareDenyWrite • adModeShareExclusive • adModeShareDenyNone
State	Περιγράφει την κατάσταση της σύνδεσης. Οι τιμές της μπορεί να είναι adStateClosed και adStateOpen

Πίνακας 16 Ιδιότητες του αντικειμένου Connection

Μέθοδος	
Close	«Κλείνει» τη σύνδεση με τη βάση δεδομένων
Open	«Ανοίγει» τη σύνδεση με τη βάση δεδομένων

Πίνακας 17 Μέθοδοι του αντικειμένου Connection

Στους επόμενους πίνακες παρατίθενται οι παράμετροι για σύνδεση με διάφορες βάσεις δεδομένων.

Παράμετρος	**Τιμή**
Provider	Microsoft.Jet.OLEDB.4.0
Data Source	Όνομα του αρχείου της βάσης δεδομένων (περιλαμβανομένης και της διαδρομής)
User ID	Όνομα χρήστη (προαιρετικό)
Password	Συνθηματικό χρήστη (προαιρετικό)

Πίνακας 18 Παράμετροι για τη σύνδεση με βάση δεδομένων Access

Παράμετρος	Τιμή
Provider	SQLOLEDB ή SQLOLEDB.1
Data Source	Όνομα του υπολογιστή (όπως εμφανίζεται ο υπολογιστής στο δίκτυο) που είναι εγκατεστημένος ο SQL Server
Initial Catalog	Όνομα της βάσης δεδομένων στον SQL Server
Integrated Sequrity	SSPI (αν χρησιμοποιούμε Windows NT/2000/2003/XP authentication)
User ID	Όνομα χρήστη (αν δεν χρησιμοποιούμε Windows NT/2000/2003/XP authentication)
Password	Συνθηματικό χρήστη (αν δεν χρησιμοποιούμε Windows NT/2000/2003/XP authentication)
Prompt	Prompt ή Complete (αν θέλουμε να εμφανιστεί παράθυρο login)

Πίνακας 19 Παράμετροι για τη σύνδεση με βάση δεδομένων SQL Server

Στο επόμενο παράδειγμα συνδεόμαστε στη βάση δεδομένων που ονομάζεται SQL_DB, βρίσκεται στον υπολογιστή SQL_PC και χρησιμοποιούμε NT authentication:

```
Dim conn As New Connection
conn.Open "Provider=SQLOLEDB;Data Source=SQL_PC;" & _
  "Initial Catalog=SQL_DB;Integrated Security=SSPI"
```

Αντίθετα στο επόμενο παράδειγμα εμφανίζεται ένα παράθυρο όπου ο χρήστης πρέπει να εισάγει όνομα και συνθηματικό:

```
Dim conn As New Connection
conn.Open "Provider=SQLOLEDB;Data Source=SQL_PC;" & _
  "Initial Catalog=SQL_DB;Integrated Security= Complete"
```

Παράμετρος	Τιμή
Provider	MSDASQL
DNS	Όνομα της πηγής δεδομένων ODBC
Driver	Όνομα του οδηγού ODBC (πχ SQL Server)
Server	Όνομα του υπολογιστή (όπως εμφανίζεται ο υπολογιστής στο δίκτυο) που είναι εγκατεστημένος ο database server
Database	Όνομα της βάσης δεδομένων στον database server
Trusted Connection	yes (αν χρησιμοποιούμε Windows NT/2000/2003/XP authentication)
UID	Όνομα χρήστη (αν δεν χρησιμοποιούμε Windows NT/2000/2003/XP authentication)
PWD	Συνθηματικό χρήστη (αν δεν χρησιμοποιούμε Windows NT/2000/2003/XP authentication)

Πίνακας 20 Παράμετροι για τη σύνδεση με βάση δεδομένων μέσω ODBC

Το αντικείμενο Recordset

Προκειμένου να διαχειριστούμε το σύνολο των δεδομένων τα οποία επιθυμούμε να ανακτήσουμε, να τροποποιήσουμε, να εισάγουμε ή να διαγράψουμε μέσω ενός ερωτήματος, χρησιμοποιούμε το αντικείμενο **Recordset**. Αυτό το αντικείμενο περιέχει το σύνολο των εγγραφών που επιστρέφονται, όταν εκτελείται μία εντολή στη βάση δεδομένων. Χρησιμοποιούμε τη μέθοδο **Open** όπως και με το αντικείμενο **Connection**, για να «ανοίξουμε» το αντικείμενο **Recordset**, όπως παρακάτω:

recst.Open source, activeconnection [, cursortype [, locktype]]

όπου:

- **source**: (υποχρεωτικά) μία εντολή SQL, ή ένα αντικείμενο **Command**, ή το όνομα ενός πίνακα, ή ένα URL, ή μία αποθηκευμένη στη βάση δεδομένων διαδικασία (stored procedure)
- **activeconnection**: (υποχρεωτικά) το ενεργό αντικείμενο **Connection** που έχουμε προηγουμένως ανοίξει
- **cursortype**: (προαιρετικά) παράμετρος που καθορίζει το τρόπο διαχείρισης των δεδομένων από τη βιβλιοθήκη ADO και τη βάση δεδομένων
- **locktype**: (προαιρετικά) παράμετρος που καθορίζει αν τα δεδομένα μπορούν να τροποποιηθούν και τι θα συμβεί σε περίπτωση διένεξης (παραπάνω από ένας χρήστες προσπαθήσουν ταυτόχρονα να τροποποιήσουν τα ίδια δεδομένα)

Στους επόμενους πίνακες δίνονται οι τιμές των παραμέτρων **cursortype** και **locktype**:

Τιμή Παραμέτρου	
adLockReadOnly	Τα δεδομένα δεν μπορούν να τροποποιηθούν
adLockPessimistic	Κλειδώνει τα δεδομένα κατά την πρόσβαση (πριν καν γίνει προσπάθεια τροποποίησης)
adLockOptimistic	Κλειδώνει τα δεδομένα κατά την τροποποίηση (μετά την πρόσβαση στα δεδομένα)
adLockBatchOptimistic	Πάνω από ένα σύνολο δεδομένων μπορεί να τροποποιηθεί ταυτόχρονα

Πίνακας 21 Τιμές της παραμέτρου locktype

Τιμή Παραμέτρου	
adOpenForwardOnly	Ευκολότερη διαχείριση αλλά και λιγότερες διαθέσιμες συναρτήσεις για το χειρισμό των δεδομένων
adOpenDynamic	Εύκολη διαχείριση με περισσότερες δυνατότητες πλοήγησης μεταξύ των δεδομένων
adOpenKeyset	Δύσκολη διαχείριση και όλες οι συναρτήσεις διαθέσιμες
adOpenStatic	Δυσκολότερη διαχείριση, αργή πρόσβαση στα πρώτα δεδομένα αλλά ταχύτερα στα επόμενα

Πίνακας 22 Τιμές της παραμέτρου cursortype

Από τη στιγμή που το αντικείμενο **Recordset** ανοιχτεί δεν είναι δυνατή η τροποποίηση των τιμών των παραμέτρων **cursortype** και **locktype**.

Στους επόμενους πίνακες παρουσιάζονται οι κυριότερες μέθοδοι και οι ιδιότητες του αντικειμένου **Recordset**:

Ιδιότητα	
ActiveConnection	Αναφέρεται σε ένα αντικείμενο Connection, που θα χρησιμοποιηθεί για τη σύνδεση με τη βάση δεδομένων
BOF	Επιστρέφει τιμή **True** όταν ο τρέχον δείκτης εγγραφής (current record pointer) δείχνει πριν τη πρώτη εγγραφή του **Recordset**
EditMode	Δηλώνει την κατάσταση της τρέχουσας εγγραφής. Μπορεί να έχει τις τιμές: • adEditNone, όταν δεν έχει τροποποιηθεί η εγγραφή • adEditInProgress, όταν έχει γίνει τροποποίηση αλλά δεν έχει αποθηκευτεί • adEditAdd, όταν με τη μέθοδο **AddNew** έχουμε δημιουργήσει μία νέα κενή εγγραφή αλλά δεν την έχουμε αποθηκεύσει • adEditDelete, όταν η τρέχουσα εγγραφή έχει διαγραφεί
EOF	Επιστρέφει τιμή **True** όταν ο τρέχον δείκτης εγγραφής δείχνει μετά την τελευταία εγγραφή του **Recordset**
Fields	Συλλογή που περιέχει τα αντικείμενα **Field** που απαρτίζουν το **Recordset**
Filter	Έκφραση παρόμοια με μία SQL δήλωση της μορφής Where ... (σταθερά ή μεταβλητή τύπου **string**)
RecordCount	Ο συνολικός αριθμός των εγγραφών που επιστρέφονται
Sort	Περιέχει τα ονόματα των στηλών πινάκων βάσει των οποίων θα ταξινομηθούν οι εγγραφές
Source	Αναφέρεται σε ένα αντικείμενο **Command** ή ένα αλφαριθμητικό που περιέχει μία εντολή SQL

Πίνακας 23 Ιδιότητες του αντικειμένου Recordset

Μέθοδος	
AddNew	Προσθέτει μετά την τελευταία εγγραφή του **Recordset**, μία νέα εγγραφή
CancelUpdate	Ακυρώνει τυχόν αλλαγές της τρέχουσας εγγραφής
Close	«Κλείνει» το αντικείμενο **Recordset**
Delete	Διαγράφει την τρέχουσα εγγραφή
Move(NumRecords)	Μετακινεί τον τρέχοντα δείκτη εγγραφής (current record pointer) τόσες θέσεις μπροστά ή πίσω όσες δηλώνει η τιμή NumRecords
MoveFirst	Μετακινεί τον τρέχοντα δείκτη εγγραφής στη πρώτη εγγραφή
MoveLast	Μετακινεί τον τρέχοντα δείκτη εγγραφής στην τελευταία εγγραφή
MoveNext	Μετακινεί τον τρέχοντα δείκτη εγγραφής στην επόμενη εγγραφή
MovePrevious	Μετακινεί τον τρέχοντα δείκτη εγγραφής στη προηγούμενη εγγραφή
Open	«Ανοίγει» ένα αντικείμενο **Recordset**, παρέχει δηλαδή πρόσβαση στα δεδομένα του **Recordset**
Update	Αποθηκεύει τις αλλαγές που έχουν γίνει στην τρέχουσα εγγραφή, στη βάση δεδομένων

Πίνακας 24 Μέθοδοι του αντικειμένου Recordset

Ανάκτηση δεδομένων με SQL ερώτημα

Στο επόμενο παράδειγμα συνδεόμαστε στη βάση δεδομένων Access northwind.mdb που δημιουργείται κατά την εγκατάσταση του Office 2003 και ανακτούμε δεδομένα εκτελόντας ένα ερώτημα μέσω μίας εντολής SQL που γράφουμε σαν μία μεταβλητή τύπου **string**. Παρατηρούμε ότι δεν ανοίγουμε πρώτα ένα **Connection** και μετά το **Recordset**, αλλά όταν ανοίγουμε το **Recordset** περνάμε με μία μεταβλητή τύπου **string** τις παραμέτρους της σύνδεσης σε αυτό απευθείας:

```
Sub SQLTextQuery()

    Dim recst As New Recordset
    Dim query As string
    Const conn As String = _
    "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=C:\Program Files\Microsoft " & _
    "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security Info=False"

    ' Δήλωση της εντολής SQL
    query = "SELECT CompanyName, ContactName " & _
    "FROM Customers " & _
    "WHERE Country = 'UK' " & _
    "ORDER BY CompanyName"

    ' Αρχικοποίηση του Recordset και εκτέλεση του ερωτήματος
    recst.Open query, conn, CursorTypeEnum.adOpenForwardOnly, _
    LockTypeEnum.adLockReadOnly, CommandTypeEnum.adCmdText

    ' Επιβεβαίωση της επιστροφής εγγραφών
    If Not recst.EOF Then
```

```
        ' Αντιγραφή των δεδομένων που περιέχει το Recordset στο φύλλο εργασίας
        Call Sheet1.Range("A2").CopyFromRecordset(recst)
        ' Προσθήκη επικεφαλίδων
        n = 0
        For Each Field In recst.Fields
            n = n + 1
            Cells(1, n) = Field.Name
        Next
    Else
        Call MsgBox("Δεν υπαρχουν εγγραφές.", vbCritical)
    End If

    ' Κλείνουμε το Recordset εφόσον είναι ανοικτό ακόμη
    If (recst.State And ObjectStateEnum.adStateOpen) Then recst.Close
    Set recst = Nothing
    Sheet1.UsedRange.EntireColumn.AutoFit

End Sub
```

	A	B	C
1	CompanyName	ContactName	
2	Around the Horn	Thomas Hardy	
3	B's Beverages	Victoria Ashworth	
4	Consolidated Holdings	Elizabeth Brown	
5	Eastern Connection	Ann Devon	
6	Island Trading	Helen Bennett	
7	North/South	Simon Crowther	
8	Seven Seas Imports	Hari Kumar	
9			
10			

Εικόνα 77 Τα αποτελέσματα που επέστρεψε το ερώτημα SQL από τη βάση δεδομένων nothwind.mdb

Ανάκτηση δεδομένων μέσω αποθηκευμένου ερωτήματος

Η Access παρέχει τη δυνατότητα δημιουργίας και αποθήκευσης ερωτημάτων. Μπορούμε να ανακτήσουμε δεδομένα από μία βάση Access χρησιμοποιώντας αποθηκευμένα ερωτήματα, όπως παρακάτω:

```
Sub StoredQuery()

    Dim recst As New Recordset
    Const conn As String = _
    "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=C:\Program Files\Microsoft " & _
    "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security Info=False"

    ' Αρχικοποίηση του Recordset και εκτέλεση του ερωτήματος
    recst.Open "[Sales By Category]", conn, _
    CursorTypeEnum.adOpenForwardOnly, _
    LockTypeEnum.adLockReadOnly, CommandTypeEnum.adCmdTable

    ' Επιβεβαίωση της επιστροφής εγγραφών
    If Not recst.EOF Then
```

```
        ' Αντιγραφή των δεδομένων που περιέχει το Recordset στο φύλλο εργασίας
        Call Sheet1.Range("A2").CopyFromRecordset(recst)
        ' Προσθήκη επικεφαλίδων
        n = 0
        For Each Field In recst.Fields
            n = n + 1
            Cells(1, n) = Field.Name
        Next
    Else
        Call MsgBox("Δεν υπαρχουν εγγραφές.", vbCritical)
    End If

    ' Κλείνουμε το Recordset εφόσον είναι ανοικτό ακόμη
    If (recst.State And ObjectStateEnum.adStateOpen) Then recst.Close
    Set recst = Nothing
    Sheet1.UsedRange.EntireColumn.AutoFit
End Sub
```

Εδώ δεν περάσαμε στο **Recordset** ένα ερώτημα με εντολή SQL, αλλά απλώς δηλώσαμε το όνομα του αποθηκευμένου ερωτήματος (**"[Sales By Category]"**) και επίσης δηλώσαμε ότι το ερώτημα είναι τύπου πίνακα (**CommandTypeEnum.adCmdTable**). Στην επόμενη εικόνα βλέπουμε μέρος των εγγραφών που επέστρεψε το αποθηκευμένο ερώτημα:

	A	B	C	D	E
1	CategoryID	CategoryName	ProductName	ProductSales	
2	1	Beverages	Chai	4887	
3	1	Beverages	Chang	7038.55	
4	1	Beverages	Chartreuse verte	4475.7	
5	1	Beverages	Côte de Blaye	49198.08	
6	1	Beverages	Guaraná Fantástica	1630.12	
7	1	Beverages	Ipoh Coffee	11069.9	
8	1	Beverages	Lakkalikööri	7379.1	
9	1	Beverages	Laughing Lumberjack Lager	910	
10	1	Beverages	Outback Lager	5468.4	
11	1	Beverages	Rhonbrau Klosterbier	4485.54	
12	1	Beverages	Sasquatch Ale	2107	
13	1	Beverages	Steeleye Stout	5274.9	
14	2	Condiments	Aniseed Syrup	1724	
15	2	Condiments	Chef Anton's Cajun Seasoning	5214.88	
16	2	Condiments	Chef Anton's Gumbo Mix	373.62	
17	2	Condiments	Genen Shouyu	1474.82	
18	2	Condiments	Grandma's Boysenberry Spread	2500	
19	2	Condiments	Gula Malacca	6737.93	
20	2	Condiments	Louisiana Fiery Hot Pepper Sauce	9373.18	
21	2	Condiments	Louisiana Hot Spiced Okra	2958	
22	2	Condiments	Northwoods Cranberry Sauce	4260	
23	2	Condiments	Original Frankfurter grüne Soße	4761.38	
24	2	Condiments	Sirop d'érable	9091.5	
25	2	Condiments	Vegie-spread	6899.25	
26	3	Confections	Chocolade	1282.01	
27	3	Confections	Gumbar Gummibärchen	10443.06	
28	3	Confections	Maxilaku	3128.6	
29	3	Confections	NuNuCa Nuß-Nougat-Creme	1692.6	
30	3	Confections	Pavlova	8663.4	
31	3	Confections	Schoggi Schokolade	10974	
32	3	Confections	Scottish Longbreads	4157.5	
33	3	Confections	Sir Rodney's Marmalade	7314.3	
34	3	Confections	Sir Rodney's Scones	5273	
35	3	Confections	Tarte au sucre	21638.29	
36	3	Confections	Teatime Chocolate Biscuits	2986.75	
37	3	Confections	Valkoinen suklaa	2173.44	
38	3	Confections	Zaanse koeken	2930.75	
39	4	Dairy Products	Camembert Pierrot	20505.4	
40	4	Dairy Products	Fløtemysost	8438.74	
41	4	Dairy Products	Geitost	786	
42	4	Dairy Products	Gorgonzola Telino	7300.74	
43	4	Dairy Products	Gudbrandsdalsost	13062.6	
44	4	Dairy Products	Mascarpone Fabioli	2668.8	
45	4	Dairy Products	Mozzarella di Giovanni	11602.8	
46	4	Dairy Products	Queso Cabrales	6911.94	
47	4	Dairy Products	Queso Manchego La Pastora	8335.3	
48	4	Dairy Products	Raclette Courdavault	35775.3	

Εικόνα 78 Τα αποτελέσματα που επέστρεψε το αποθηκευμένο ερώτημα από τη βάση δεδομένων nothwind.mdb

Πεδία δεδομένων

Κάθε εγγραφή δεδομένων ενός αντικειμένου **Recordset** περιέχει ένα ή περισσότερα πεδία δεδομένων (**Data Fields**). Η συλλογή **Fields** περιέχει όλα τα πεδία ενός **Recordset**. Αναφερόμαστε σε ένα πεδίο ενός **Recordset** ως εξής:

recordsetname!fieldname

όπου:

- **recordsetname**: το όνομα του αντικειμένου **Recordset**
- **fieldname**: το όνομα του αντικειμένου **Field**

Στο επόμενο παράδειγμα συνδεόμαστε στη βάση δεδομένων Access northwind.mdb και ανακτούμε τα δεδομένα του πίνακα employee εκτελώντας και στη συνέχεια γράφουμε στις τρεις πρώτες στήλες του φύλλου εργασίας Sheet1, τα ονόματα των πεδίων του πίνακα employee, τον τύπο τους και όνομα του τύπου τους.

```
Sub RecFields()

   Dim conn As New Connection
   Dim recst As New Recordset, f As Field
   Dim ws As Worksheet, i&

   Set ws = ThisWorkbook.Worksheets("Sheet1")

   conn.Open "Provider=microsoft.jet.oledb.4.0;" & _
     " Data Source=C:\Program Files\Microsoft " & _
     "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security Info=False"

   recst.Open "employees", conn

   n = 0
   For Each fld In recst.Fields
      n = n + 1
      ws.[a1].Cells(n) = fld.Name
      ws.[b1].Cells(n) = TypeName(fld.Value)
   Next

   ws.UsedRange.EntireColumn.AutoFit
   recst.Close
   conn.Close

End Sub
```

Το αποτέλεσμα του έχει η εκτέλεση του παραπάνω κώδικα φαίνεται στην επόμενη εικόνα:

	A	B	C
1	EmployeeID	Long	
2	LastName	String	
3	FirstName	String	
4	Title	String	
5	TitleOfCourtesy	String	
6	BirthDate	Date	
7	HireDate	Date	
8	Address	String	
9	City	String	
10	Region	String	
11	PostalCode	String	
12	Country	String	
13	HomePhone	String	
14	Extension	String	
15	Photo	String	
16	Notes	String	
17	ReportsTo	Long	
18			

Εικόνα 79 Τα πεδία (όνομα και τύπος) του πίνακα employee

Μετακίνηση μεταξύ εγγραφών, τροποποίηση τιμών πεδίων, προσθήκη και διαγραφή εγγραφών

Όταν ένα αντικείμενο **Recordset** ανοίγει για πρώτη φορά, ο τρέχον δείκτης εγγραφής βρίσκεται στην πρώτη εγγραφή. Μπορούμε να μετακινήσουμε το δείκτη χρησιμοποιώντας τις μεθόδους **MoveFirst**, **MoveLast**, **MovePrevious** ή **MoveNext**, όπως στο παρακάτω παράδειγμα:

```
Sub MovePointer()

    Dim conn As New Connection
    Dim recst As New Recordset, f As Field
    Dim ws As Worksheet, i&

    conn.Open "Provider=microsoft.jet.oledb.4.0;" & _
      "Data Source=C:\Program Files\Microsoft " & _
      "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security Info=False"

    recst.Open "employees", conn

    recst.MoveFirst
    ' Ελέγχουμε αν είμαστε στην τελευταία εγγραφή
    While Not recst.EOF
        Debug.Print recst.Fields("FirstName") & " " & recst.Fields("LastName")
        ' Μετακινούμαστε στην επόμενη εγγραφή του recordset
        recst.MoveNext
    Wend

    recst.Close
    conn.Close

End Sub
```

Προκειμένου να τροποποιήσουμε την τιμή ενός πεδίου μίας εγγραφής (όπως επίσης και αν θέλουμε να προσθέσουμε ή να διαγράψουμε εγγραφές), πρέπει να δώσουμε την τιμή **adLockOptimistic** στην παράμετρο **locktype**. Η τροποποίηση γίνεται ως εξής:

recordsetname!fieldname = newvalue

Αν το όνομα του πεδίου περιέχει κενά ή ειδικούς χαρακτήρες χρησιμοποιούμε αγκύλες (**[fieldname]**). Μετά την αλλαγή τιμής ενός ή περισσότερων πεδίων πρέπει να χρησιμοποιήσουμε τη μέθοδο **update**, προκειμένου να αποθηκεύσουμε την αλλαγή. Στο επόμενο παράδειγμα τροποποιούμε την τιμή του πεδίου FirstName της πρώτης εγγραφής και χρησιμοποιούμε το παράθυρο immediate για να δούμε την αλλαγή.

```
Sub EditRecord()

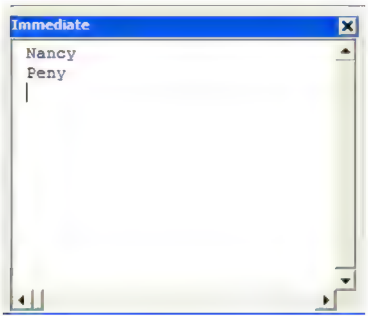
   Dim conn As New Connection
   Dim recst As New Recordset, f As Field
   Dim ws As Worksheet, i&

   conn.Open "Provider=microsoft.jet.oledb.4.0;" & _
     " Data Source=C:\Program Files\Microsoft " & _
     "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security Info=False"

   recst.Open "employees", conn , adOpenForwardOnly, adLockOptimistic

   recst.MoveFirst
   Debug.Print recst!FirstName
   ' Τροποποιούμε την τιμή του πεδίου FirstName της πρώτης εγγραφής
   recst!FirstName = "Peny"
   Debug.Print recst!FirstName
   ' Επικυρώνουμε την τροποποίηση
   recst.Update
   recst.Close
   conn.Close

End Sub
```



Εικόνα 80 Το αποτέλεσμα του κώδικα στο παράθυρο immediate

Αντίστοιχα η προσθήκη νέας εγγραφής γίνεται με τη μέθοδο **AddNew** (**recordset.AddNew**) και η διαγραφή με τη μέθοδο **Delete** (**recordset.Delete**).

Προσθήκη, τροποποίηση και διαγραφή εγγραφών με SQL ερωτήματα

Μπορούμε να χρησιμοποιήσουμε εντολές SQL προκειμένου να προσθέσουμε, να τροποποιήσουμε ή να διαγράψουμε εγγραφές, από ένα **Recordset**. Σε αυτή την περίπτωση δεν απαιτείται να δώσουμε την

τιμή **adLockOptimistic** στην παράμετρο **locktype** του αντικειμένου **Recordset**. Στο παρακάτω παράδειγμα εκτελούμε τρία διαφορετικά ερωτήματα SQL και χρησιμοποιούμε το παράθυρο immediate για να δούμε τις μεταβολές στον πίνακα Shippers που επιφέρει η εκτέλεση των ερωτημάτων.

```
Sub SQLInsertUpdateDelete()

   Dim conn As New Connection
   Dim recst As New Recordset, f As Field
   Dim query As String

   conn.Open "Provider=microsoft.jet.oledb.4.0;" & _
     " Data Source=C:\Program Files\Microsoft " & _
     "Office\OFFICE11\SAMPLES\Northwind.mdb;Persist Security Info=False"

   Call display_table(recst, conn, "Original table:")

   ' Προσθέτουμε μία νέα εγγραφή
   query = "INSERT INTO Shippers(CompanyName, Phone) " & _
   "VALUES ('ICARUS', '21010099100')"
   recst.Open query, conn
   Call display_table(recst, conn, "New recorder added:")

   ' Τροποποιούμε ένα πεδίο της νέας εγγραφής
   query = "UPDATE Shippers SET Phone ='21010099101'" & _
   "WHERE CompanyName='ICARUS'"
   recst.Open query, conn
   Call display_table(recst, conn, "New record altered:")

   ' Διαγράφουμε την εγγραφή που προσθέσαμε
   query = "DELETE FROM Shippers " & _
   "WHERE CompanyName='ICARUS'"
   recst.Open query, conn
   Call display_table(recst, conn, "New record deleted:")

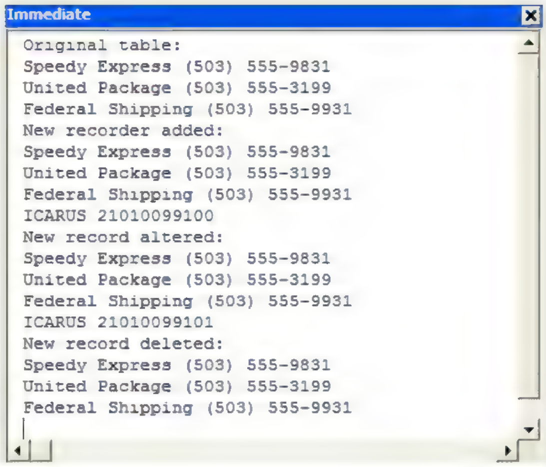
   conn.Close

End Sub
```

```
Sub display_table(recst As Recordset, conn As Connection, query_command As
String)

   Debug.Print query_command
   recst.Open "Shippers", conn
   recst.MoveFirst
   While Not recst.EOF
     Debug.Print recst.Fields("CompanyName") & " " & recst.Fields("Phone")
      recst.MoveNext
   Wend
   recst.Close

End Sub
```



Εικόνα 81 Οι μεταβολές των εγγραφών του πίνακα Shippers

Σύντομη περιγραφή της γλώσσας SQL

Η γλώσσα SQL (Structured Query Language) χρησιμοποιείται στις περισσότερες σύγχρονες βάσεις δεδομένων. Κάθε κατασκευαστής βάσεων δεδομένων (Microsoft, Orracle κλπ) χρησιμοποιεί μία παραλλαγή της, που επεκτείνει το πρότυπο (standard) SQL-92. Παρακάτω παρουσιάζουμε τις βασικές εντολές και τη σύνταξή τους.

Εντολή	Περιγραφή
SELECT	Επιλέγει ένα ή περισσότερα πεδία από ένα ή περισσότερους πίνακες
INSERT	Εισάγει νέες εγγραφές σε ένα πίνακα
UPDATE	Τροποποιεί τιμές πεδίων σε εγγραφές ενός πίνακα
DELETE	Διαγράφει εγγραφές από ένα πίνακα
CREATE TABLE	Δημιουργεί ένα νέο πίνακα
DROP TABLE	Διαγράφει ένα πίνακα

Πίνακας 25 Βασικές εντολές SQL

Οι τέσσερις πρώτες εντολές συνδυάζονται πολύ συχνά με την πρόταση (clause) WHERE προκειμένου να περιοριστεί ο αριθμός των εγγραφών που θα επιστρέψει το ερώτημα σε αυτές που πραγματικά επιθυμούμε. Επίσης οι εγγραφές μπορούν να ταξινομηθούν τη χρησιμοποιώντας την πρόταση ORDER BY. Η πρόταση GROUP BY χρησιμοποιείται για να ομαδοποιηθούν οι εγγραφές που των οποίων ένα ή περισσότερα πεδία έχουν την ίδια τιμή ή τιμές. Συνδυάζεται με συναρτήσεις άθροισης, όπως η SUM(columnname) που υπολογίζει το άθροισμα των τιμών ενός πεδίου όλων των εγγραφών που έχουν ομαδοποιηθεί. Τα πεδία που επιστρέφονται μπορούν να μετονομαστούν (συνήθως για να είναι κατανοητό το είδος δεδομένων που περιέχουν, καθότι πολλές φορές τα όνομα τους δεν είναι διαφωτιστικά). Σημειώνουμε ότι αν και οι εντολές γράφονται κατά σύμβαση με κεφαλαία, η γλώσσα SQL δεν διαχωρίζει μεταξύ πεζών και κεφαλαίων.

Βασική σύνταξη των εντολών:

```
SELECT column1[, column2,... columnN]
FROM table
[WHERE filter]
[GROUP BY column1[, column2,... columnN]]
[ORDER BY column1[, column2,... columnN]]

INSTERT INTO table (column1[, column2,... columnN])
VALUES (value1 [, value2,… valueN])
[WHERE filter]

UPDATE table SET column1=value1 [, column2=value2,… columnN=valueN]
[WHERE filter]

DELETE FROM table
```

Παραδείγματα:

1. Το ερώτημα επιστρέφει τα πεδία ProductName και UnitPrice, από τον πίνακα Products, για τις εγγραφές όπου το UnitPrice έχει τιμή μικρότερη από 20.

   ```
   SELECT ProductName, UnitPrice
   FROM Products
   WHERE UnitPrice < 20
   ORDER BY ProductName
   ```

2. Το ερώτημα επιστρέφει τα πεδίων ProductName, UnitPrice και CategoryName, από τους πίνακες Products και Categories, για τις εγγραφές όπου το UnitPrice έχει τιμή μικρότερη από 20.

   ```
   SELECT ProductName, CategoryName, UnitPrice
   FROM Products, Categories
   WHERE UnitPrice < 20 AND Categories.CategoryID = Products.CategoryID
   ORDER BY ProductName
   ```

3. Το ερώτημα μας επιστρέφει το πλήθος των εγγραφών του πίνακα Orders, και τη μέση τιμή του πεδίου Freight. Η λέξη κλειδί AS χρησιμοποιείται για να μετονομάσει τα πεδία.

   ```
   SELECT COUNT(*) AS quant, AVG(Freight) AS aver
   FROM Orders
   ```

4. Το ερώτημα υπολογίζει το άθροισμα των εσόδων από τις πωλήσεις (UnitPrice * Quantity * (1-Discount) από τις εγγραφές του πίνακα Order Details, ανά προϊόν ProductID.

   ```
   SELECT ProductID, SUM(UnitPrice * Quantity * (1-Discount)) AS Sales
   FROM [Order Details]
   GROUP BY OrderID
   ```

Παραδείγματα

Συναρτήσεις χρήστη

Η συνάρτηση split_string_left

Η συνάρτηση βρίσκει το χαρακτήρα " " και επιστρέφει το μέρος του κειμένου που βρίσκεται αριστερά του χαρακτήρα.

```
Function split_string_left(CellAddr As String) As String

    Dim space_pos As Integer

    space_pos = Application.WorksheetFunction.Find(" ", CellAddr)
    split_string_left = Left(CellAddr, space_pos – 1)

End Function
```

Χρησιμοποιούμε το αντικείμενο.ιδιότητα **Application.WorksheetFunction** για να καλέσουμε μία συνάρτηση του Excel. Εδώ καλούμε τη συνάρτηση **Find(Arg1 as String, Arg2 as String, [Arg3 as Double])** η οποία αναζητά το αλφαριθμητικό **Arg1** μέσα στο αλφαριθμητικό **Arg2** ξεκινώντας από το θέση **Arg3** και επιστρέφει τη θέση του **Arg1**.

Η συνάρτηση **Left(String, Length as Long)** της VBA επιστρέφει αριθμό χαρακτήρων ίσο με **Lenght** από το αριστερό άκρο του αλφαριθμητικού **String**. Το όνομα της συνάρτησης (**split_string_left**) χρησιμοποιείται ως μεταβλητή την οποία αποδίδεται μία τιμή που θα επιστρέψει η συνάρτηση. Το αλφαριθμητικό **split_string_left** αποδίδεται η τιμή που επιστρέφει η συνάρτηση **Left**.

Η συνάρτηση split_string_right

Η συνάρτηση βρίσκει τη θέση του χαρακτήρα " " και επιστρέφει το μέρος του κειμένου που βρίσκεται δεξιά του χαρακτήρα

```
Function split_string_right(CellAddr As String) As String

    Dim space_pos As Integer

    space_pos = Application.WorksheetFunction.Find(" ", CellAddr)
    split_string_right = Right(CellAddr, Len(CellAddr) - space_pos)

End Function
```

Η συνάρτηση βρίσκει τη θέση του χαρακτήρα " " και επιστρέφει το μέρος του κειμένου που βρίσκεται δεξιά του χαρακτήρα.

Η συνάρτηση split_string_left_sel_char

Η συνάρτηση βρίσκει τη θέση του χαρακτήρα που περνάει ως δεύτερο προαιρετικό όρισμα (**Split_Char**) και επιστρέφει το μέρος του κειμένου που βρίσκεται αριστερά του χαρακτήρα

```
Function split_string_left_sel_char(CellAddr As String, Optional Split_Char _
As String) As String

    Dim space_pos As Integer

    Debug.Print Split_Char
    If Split_Char = "" Then Split_Char = " "
    Debug.Print Split_Char
    space_pos = Application.WorksheetFunction.Find(Split_Char, CellAddr)
    split_string_left_sel_char = Left(CellAddr, space_pos - 1)

End Function
```

Η συνάρτηση split_string_left_with_error_handler

Η συνάρτηση βρίσκει τη θέση του χαρακτήρα " " και επιστρέφει το μέρος του κειμένου που βρίσκεται αριστερά του χαρακτήρα. Αν δεν βρει το χαρακτήρα " " επιστρέφει ένα μήνυμα λάθους.

```
Function split_string_left_with_error_handler(CellAddr As String) As String

    Dim space_pos As Integer

    'Αν παρουσιαστεί σφάλμα στην επόμενη εντολή η εκτέλεση μεταβαίνει
    ' στο σημείο που δηλώνεται από την ετικέτα Error_handler
    On Error GoTo Error_handler

    space_pos = Application.WorksheetFunction.Find(" ", CellAddr)
    split_string_left_with_error_handler = Left(CellAddr, space_pos - 1)
    Exit Function 'Η εκτέλεση της συνάρτησης διακόπτεται

Error_handler: 'Η ετικέτα Error_handler
    split_string_left_with_error_handler = "Προέκυψε σφάλμα κατά τον _
    υπολογισμό"

End Function
```

Η συνάρτηση show_type

Η συνάρτηση επιστρέφει τον τύπο του άνω αριστερού κελιού της περιοχής του ορίσματος.

```
Function show_type(Rng As Range)

    Application.Volatile

    Set Rng = Rng.Range("A1")

    Select Case True
        Case IsEmpty(Rng): show_type = "Κενό"
        Case Application.IsText(Rng): show_type = "Κείμενο"
        Case Application.IsLogical(Rng): show_type = "Λογική Τιμή"
        Case Application.IsErr(Rng): show_type = "Σφάλμα"
        Case IsDate(Rng): show_type = "Ημερομηνία"
        Case InStr(1, Rng.Text, ":") <> 0: show_type = "Ώρα"
        Case IsNumeric(Rng): show_type = "Αριθμός"
    End Select

End Function
```

Η έκφραση **Application.Volatile** δηλώνει ότι η τιμή της συνάρτησης πρέπει να υπολογιστεί εκ νέου όταν υπολογιστεί η τιμή οποιουδήποτε κελιού στο φύλλο εργασίας.

Η συνάρτηση tax

Η συνάρτηση υπολογίζει παρακράτηση φόρου, χρησιμοποιώντας διαφορετικά ποσοστά (**tax1** έως **tax6**) ανάλογα με το ύψος του φορολογούμενου ποσού (**amount**).

```
Function tax(amount)

    Dim tax1 As Double
    Dim tax2 As Double
    Dim tax3 As Double
    Dim tax4 As Double
    Dim tax5 As Double
    Dim tax6 As Double

    tax1 = 0.05
    tax2 = 0.1
    tax3 = 0.15
    tax4 = 0.2
    tax5 = 0.25
    tax6 = 0.3

    Select Case amount
        Case Is <= 500
            tax = amount * tax1
        Case 500.001 To 800
            tax = amount * tax2
        Case 800.0011 To 1000
            tax = amount * tax3
        Case 1000.001 To 1200
            tax = amount * tax4
        Case 1200.001 To 1500
            tax = amount * tax5
        Case Is > 1500
            tax = amount * tax6
```

```
    End Select
End Function
```

Ο αντίστοιχος τύπος του Excel, που θα πραγματοποιούσε τον ίδιο υπολογισμό θα ήταν:

=IF(B42<=500;B42*0,05;IF(AND(B42>500;B42<=800);B42*0,1;IF(AND(B42>800;B42<=1000);B42*0,15;IF(AND(B42>1000;B42<=1200);B42*0,2;IF(AND(B42>1200;B42<=1500);B42*0,25;B42*0,3)))))

Το παράδειγμα δείχνει πως μπορούμε να πετύχουμε το ίδιο αποτέλεσμα, με μία συνάρτηση χρήστη, αντί να χρησιμοποιούμε πολύπλοκους τύπους.

Διαδικασίες διαχείρισης συμβάντων

Το συμβάν SelectionChange του αντικειμένου Worksheet

Το συμβάν λαμβάνει χώρα όταν επιλέξουμε μία νεά περιοχή κελιών. Η παρακάτω διαδικασία διαχείρισης καθαρίζει όλα τα κελιά του φύλλο εργασίας από τυχόν χρώμα γεμίσματος και στη συνέχεια μορφοποιεί ολόκληρη την γραμμή και ολόκληρη τη στήλη που βρίσκεται το άνω αριστερό κελί της περιοχής που επιλέξαμε με χρώμα γεμίσματος.

```
Private Sub Worksheet_SelectionChange(ByVal Target As Excel.Range)
' Η διαδικασία καλείτια όταν επιλεγεί μία νεά περιοχή κελιών

    ' Μορφοποιούμε τα κελιά χωρίς χρώμα γεμίσματος
    ' η έκφραση cells επιστρέφει όλα τα κελιά του ενεργού φύλλου
    Cells.Interior.ColorIndex = xlNone

    With ActiveCell
        .EntireRow.Interior.ColorIndex = 24
        .EntireColumn.Interior.ColorIndex = 24
    End With

End Sub
```

Ο κώδικας αποθηκεύεται στο φύλλο που θέλουμε να ανιχνεύεται το συγκεκριμένο συμβάν.

Το συμβάν BeforeRightClick του αντικειμένου Worksheet

Το συμβάν λαμβάνει χώρα όταν κάνουμε δεξί κλικ στο φύλλο εργασίας. Η διαδικασία εμφανίζει ένα **InputBox** και ζητάει ένα κωδικό. Αν δεν εισαχθεί ο σωστός κωδικός εμφανίζεται μήνυμα και καταστέλεται η εμφάνισει του αναδυόμενου μενού (**Cancel = True**).

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Excel.Range, _
                                        Cancel As Boolean)
' η διαδικασία καλείται όταν κάνουμε δεξί κλικ
```

```
' αν Cancel = True δεν εκτελείται η προκαθορισμένη ενέργεια

    pswd = InputBox(Prompt:="Εισάγετε κωδικό")
    If Not pswd = "κωδικός" Then
        MsgBox "Δεν είναι διαθέσιμο το δεξί κλικ"
        Cancel = True
    Else
        Cancel = False
    End If

End Sub
```

Το συμβάν Change του αντικειμένου Worksheet

Το συμβαν λαμβάνει χώρα όταν υπάρξει μεταβολή της τιμής ενός κελιού σε ένα φύλλο. Η διαδικασία διαχείρισης του συμβάντος αποθηκεύει σε ένα φύλλο που ονομάζεται “Ιστορικό” όλες τις νέες τιμές των κελιών, την ημερομηνία και ώρα που γίνεται η αλλαγή και το χρήστη από τον οποίο έγινε η αλλαγή.

```
Private Sub Worksheet_Change(ByVal Target As Excel.Range)

    Dim cellddr As String
    Dim newcellvalue As Variant

    For Each cell In Target
        With cell
            On Error Resume Next
            newcellvalue = cell.Value
            celladr = cell.Address
            track_changes newcellvalue, celladr
        End With
    Next cell
    Worksheets("Sheet1").Activate

End Sub

Sub track_changes(newcellvalue, celladr)

    Dim r As Integer
    Dim wrksht As Worksheet

    r = 1
    ' Αναζητάμε το πρώτο άδειο κελί
    Do Until IsEmpty(Worksheets("Ιστορικό").Cells(r, 1))
        r = r + 1
    Loop

    If newcellvalue = "" Then newcellvalue = "Διαγραφή"

    Set wrksht = Worksheets("Ιστορικό")
    With wrksht
        .Cells(r, 1) = Now
        .Cells(r, 2) = celladr
        .Cells(r, 3) = newcellvalue
        .Cells(r, 4) = "από χρήστη"
        .Cells(r, 5) = Application.UserName
```

```
    End With

End Sub
```

Ολοκληρωμένο παράδειγμα 1

Στο επόμενο παράδειγμα χρησιμοποιούμε μία φόρμα για να επιλέξουμε μία από τις παρακάτω λειτουργίες:

1. Υπολογισμός μισθοδοσίας τρέχοντος μηνός για κάθε υπάλληλο του φύλλου "Sheet1"
2. Προεπισκόπηση εκτυπώσεων βεβαιώσεων μισθοδοσίας
3. Δημιουργία ενός εγγράφου Word με τις βεβαιώσεις
4. Διαγραφή των βεβαιώσεων

Για να δημιουργήσουμε έγγραφα του Word, και γενικότερα για να ελέγξουμε την εφαρμογή Word μέσα από κώδικα VBA στο Excel, πρέπει να έχουμε δηλώσει τη σχετική βιβλιοθήκη **Microsoft Word 11.0 Object Library,** στο μενού **Tools – References** (ανάλογα με την έκδοση του Office που έχουμε εγκατεστημένο στον υπολογιστή μας, μπορεί να αλλάζει η έκδοση της). Μέσω αυτής της βιβλιοθήκης μπορούμε να εκκινήσουμε την εφαρμογή του Word, και να εκτελέσουμε κώδικα που να ελέγχει την εφαρμογή.

Ο παρακάτω κώδικας περιέχεται στο module:

```
Sub Calculate_Working_Years()

    Dim cell As Range
    Dim bdtadr As String    ' διεύθυνση τρέχοντος κελιού
    Dim bdt As Date         ' ημερομηνία γέννησης
    Dim hdt As Date         ' ημερομηνία πρόσληψης
    Dim wmonths As Integer  ' μήνες προϋπηρεσίας
    Dim wyears As Integer   ' πλήρη έτη προϋπηρεσίας
    Dim cat As String       ' κλάδος
    Dim bsal As Integer     ' βασικός μισθός
    Dim rofst As Integer    ' σχετική μετατόπιση
    Dim mar As String       ' έγγαμος ή άγαμος
    Dim marb As Integer     ' επίδομα γάμου
    Dim cld As Integer      ' αριθμός τέκνων
    Dim cldb As Integer     ' επίδομα τέκνων
    Dim pos As String       ' θέση
    Dim posb As Integer     ' επίδομα θέσης
    Dim edtl As String      ' τίτλος πτυχίου
    Dim edtlb As Integer    ' επίδομα πτυχίου
    Dim nm As String        ' όνομα
    Dim srnm As String      ' επίθετο
    Dim sex As String       ' φύλλο

  ' εξασφαλίζουμε ότι το 1ο φύλλο που περιέχει τα δεδομένα των υπαλλήλων
```

```
' είναι ενεργό
 ThisWorkbook.Sheets(1).Activate
' για κάθε κελί εξετάζουμε αν περιέχει ημερομηνία και υπολογίζουμε
' τις παραμέτρους
 For Each cell In Columns("C").Cells.SpecialCells(xlCellTypeConstants)
    If cell.Value Like "*/*/*" Then

        ' η διεύθυνση του τρέχοντος κελιού
        bdtadr = cell.Address

        ' χρησιμοποιούμε θέσεις σχετικές με το τρέχον κελί
        ' για να ανακτήσουμε κάθε φορά τα στοιχεία που επιθυμούμε
        nm = Range(bdtadr).Offset(columnOffset:=-1).Value
        srnm = Range(bdtadr).Offset(columnOffset:=-2).Value
        sex = Range(bdtadr).Offset(columnOffset:=8).Value
        bdt = Range(bdtadr).Value
        hdt = Range(bdtadr).Offset(columnOffset:=1).Value

        ' η συνάρτηση datdiff επιστρέφει τη διαφορά δύο ημερομηνιών,
        ' σε μορφή που δηλώνεται από το 1ο όρισμα (εδώ μήνες: "m")
        ' για να υπολογίσουμε τον χρόνο προϋπηρεσίας
        wmonths = DateDiff("m", hdt, Date)
        ' η συνάρτηση int επιστρέφει το ακέραιο μέρος ενός αριθμού
        wyears = Int(wmonths / 12)
        cat = Range(bdtadr).Offset(columnOffset:=2).Value

        ' υπολογίζουμε τη σχετική θέση που θα βρίσκεται το κλιμάκιο
        ' ανάλογα με τα έτη προϋπηρεσίας
        rofst = Int((wyears - 1) / 2)

        ' ανάλογα με τον κλάδο επιλέγουμε τον βασικό μισθό
        Select Case cat
           Case "YE"
              bsal = ThisWorkbook.Sheets(2).Range("C3").Offset(rowoffset:=rofst)
           Case "ΔΕ"
              bsal = ThisWorkbook.Sheets(2).Range("C3").Offset _
              (rowoffset:=rofst,  columnOffset:=1)
           Case "ΠΕ"
              bsal = ThisWorkbook.Sheets(2).Range("C3").Offset _
              (rowoffset:=rofst, columnOffset:=2)
        End Select

        mar = Range(bdtadr).Offset(columnOffset:=4).Value
        If mar = "NAI" Then
           marb = ThisWorkbook.Sheets(2).Range("H3")
        Else
           marb = 0
        End If

        cld = Range(bdtadr).Offset(columnOffset:=5).Value
        ' το συνολικό επίδομα τέκνων ισούται με το γινόμενο του αριθμού
        ' των τέκων επί το επίδομα
        cldb = ThisWorkbook.Sheets(2).Range("I3") * cld

        pos = Range(bdtadr).Offset(columnOffset:=6).Value
```

```
        ' ανάλογα με τη θέση υπολογίζεται και το σχετικό επίδομα
        Select Case pos
           Case "Y"
              posb = 0
           Case "ΠΤ"
              posb = ThisWorkbook.Sheets(2).Range("L3")
           Case "ΠΔ"
              posb = ThisWorkbook.Sheets(2).Range("M3")
        End Select

        edtl = Range(bdtadr).Offset(columnOffset:=7).Value
        ' ανάλογα με το τίτλο σπουδών υπολογίζεται το σχετικό επίδομα
        Select Case edtl
           Case "B"
              edtlb = 0
           Case "M"
              edtlb = ThisWorkbook.Sheets(2).Range("J3")
           Case "Δ"
              edtlb = ThisWorkbook.Sheets(2).Range("K3")
        End Select

        ' καλούμε την υπορουτίνα που εισάγει ένα φύλλο για κάθε υπάλληλο
        Insert_Sheet cat, wyears, bsal, marb, cldb, posb, edtlb, nm, srnm, sex
        ' ενεργοποιούμε ξανά το φύλλο με τα δεδομένα μας
        ThisWorkbook.Sheets(1).Activate

     End If

  Next cell

End Sub
```

```
Sub Insert_Sheet(cat As String, wyears As Integer, bsal As Integer, marb As Integer, cldb As Integer, posb As Integer, edtlb As Integer, nm As String, srnm As String, sex As String)

  Dim txt As String
  Dim sxartcl As String      ' άρθρο ανάλογα με το φύλλο του υπαλλήλου
  Dim tot As Integer         ' σύνολο μεικτών αποδοχών

  ' προσθέτουμε ένα νέο φύλλο μετά το τελευταίο φύλλο
  ThisWorkbook.Sheets.Add after:=Worksheets(ThisWorkbook.Sheets.Count)
  ' δίνουμε το ονοματεπώνυμο του υπαλλήλου ως όνομα στο φύλλο
  Worksheets(ThisWorkbook.Sheets.Count).Name = srnm & " " & nm

  tot = bsal + marb + cldb + posb + edtlb

  Cells(2, 2) = "ΒΕΒΑΙΩΣΗ ΜΙΣΘΟΔΟΣΙΑΣ"
  Cells(4, 2) = "ΕΤΟΣ : " & Year(Date)
  ' καλούμε τη συνάρτηση που εξάγει το μήνα από την τρέχουσα ημερομηνία
  Cells(5, 2) = "ΜΗΝΑΣ : " & Find_Month

  ' επιλέγουμε άρθρο ανάλογα με το φύλλο του υπαλλήλου
```

```
    If sex = "A" Then
        sxartcl = "o"
    Else
        sxartcl = "η"
    End If

    ' εισάγουμε τα στοιχεία μισθοδοσίας στο νέο φύλλο
    txt = "Βεβαιώνουμε ότι " & sxartcl & " " & nm & " " & srnm
    txt = txt & " έλαβε μεικτές αποδοχές  "

    Cells(7, 2) = txt
    Cells(7, 9) = tot

    Cells(9, 2) = "Βασικός μισθός"
    Cells(9, 9) = bsal
    Cells(10, 2) = "Επίδομα γάμου"
    Cells(10, 9) = marb
    Cells(11, 2) = "Επίδομα τέκνων"
    Cells(11, 9) = cldb
    Cells(12, 2) = "Επίδομα θέσης"
    Cells(12, 9) = posb
    Cells(13, 2) = "Επίδομα μεταπτυχιακού τίτλου"
    Cells(13, 9) = edtlb

    ' μορφοποιούμε κατάλληλα τα κελιά που περιέχουν ποσά
    Range("I7").NumberFormat = "#,##0.00 $"
    Range("I7:I13").NumberFormat = "#,##0.00 $"

End Sub
```

```
Function Find_Month() As String

    Dim cdt As String
    Dim mnt As Integer, mnt2 As Integer, mnt3 As Integer

    ' η μεταβλητή cdt περιέχει την τρέχουσα ημερομηνάι στη μορφή
    ' "Κυριακή , 13 Απρίλης 2008"
    cdt = FormatDateTime(Date, vbLongDate)
    ' αναζητούμε τη θέση όπου υπάρχει ", "
    mnt = Application.WorksheetFunction.Find(", ", cdt)
    ' αναζητούμε τη θέση όπου υπάρχει το δεύτερο διάστημα
    mnt2 = Application.WorksheetFunction.Find(" ", cdt, mnt + 2)
    ' αναζητούμε τη θέση όπου υπάρχει το τρίτο διάστημα
    mnt3 = Application.WorksheetFunction.Find(" ", cdt, mnt2 + 1)
    ' επιστρέφουμε το μήνα
    Find_Month = Mid(cdt, mnt2 + 1, mnt3 - mnt2)

End Function
```

```
Sub Print_Sheets()

    Dim wrksht As Worksheet
```

```
    ' ελέγχουμε αν υπάρχουν φύλλα με βεβαιώσεις
    If ThisWorkbook.Sheets.Count = 2 Then
        MsgBox "Δεν υπάρχουν βεβαιώσεις για εκτύπωση"
    Else
        ' για κάθε φύλλο μετά το 2ο εμφανίζουμε μάι προεπισκόπιση εκτύπωσης
        For Each wrksht In ThisWorkbook.Sheets
            If wrksht.Index > 2 Then wrksht.PrintPreview
        Next wrksht
    End If

End Sub
```

```
Sub Delete_Sheets()

    Dim wrksht As Worksheet

    ' καταστέλουμε τυχόν μηνύμα προειδοποίησης
    Application.DisplayAlerts = False

    ' ελέγχουμε αν υπάρχουν φύλλα με βεβαιώσεις
    If ThisWorkbook.Sheets.Count = 2 Then
        MsgBox "Δεν υπάρχουν βεβαιώσεις για διαγραφή"
    Else
    ' διαγράφουμε όλα τα φύλλα που προσθέσαμε
        For Each wrksht In ThisWorkbook.Sheets
            If wrksht.Index > 2 Then wrksht.Delete
        Next wrksht
    End If

    ' ενεργοποιούμε τα μηνύματα προειδοποίησης
    Application.DisplayAlerts = True

End Sub
```

```
Sub Create_Word_Document()

    Dim wrdapp As New Word.Application
    Dim wrksht As Worksheet

    ' ελέγχουμε αν υπάρχουν φύλλα με βεβαιώσεις
    If ThisWorkbook.Sheets.Count = 2 Then
        MsgBox "Δεν υπάρχουν βεβαιώσεις για έγγραφο Word"
    Else
    ' δημιουργούμε ένα αντικείμενο της εφαρμογής word
        On Error Resume Next
        Set wrdapp = GetObject(, "Word.Application")

        If wrdapp Is Nothing Then
            Set wrdapp = GetObject("", "Word.Application")
        End If
```

```
        On Error GoTo 0

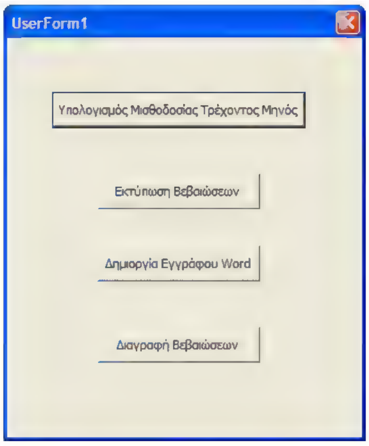
        ' προσθέτουμε ένα νέο έγγραφο και το κάνουμε ορατό
        With wrdapp
            .Documents.Add
            .Visible = True
        End With

        ' για κάθε φύλλο που προσθέσαμε ατνιγράφουμε την περιοχή κελιών
        ' που περιέχει τα στοιχεία μισθοδοσίας
        ' και την επικολούμε στο έγγραφο του word
        ' και στη συνέχεια προσθέτουμε μία αλλαγή σελίδας
        For Each wrksht In ThisWorkbook.Sheets
            If wrksht.Index > 2 Then
                wrksht.Range("B2:I13").Copy
                With wrdapp.Selection
                    .Paste
                    If wrksht.Index < ThisWorkbook.Sheets.Count Then
                        .InsertBreak Type:=wdPageBreak
                    End If
                End With
            End If
        Next wrksht

        ' εμφανίζουμε μία προεπισκόπηση εκτύπωσης του ενεργού εγγράφου
        wrdapp.ActiveDocument.PrintPreview

    End If
```

Η φόρμα φαίνεται στην επόμενη εικόνα:



Εικόνα 82 Η φόρμα που καλεί τις διαδικασίες

Ο κώδικας της φόρμας είναι ο παρακάτω:

```
Private Sub CommandButton1_Click()

   Calculate_Working_Years

End Sub
```

```
Private Sub CommandButton2_Click()

   Print_Sheets

End Sub
```

```
Private Sub CommandButton3_Click()

   Delete_Sheets

End Sub
```

```
Private Sub CommandButton4_Click()

   Create_Word_Document

End Sub
```

Στο βιβλίο εργασίας υπάρχουν δύο διαδικασίες. Η πρώτη εξασφαλίζει ότι θα διαγραφούν οι βεβαιώσεις όταν ξακλείσουμε το βιβλίο και η δεύτερη εμφανίζει τη φόρμα όταν ανοίγουμε το βιβλίο.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)

   ' ιαγράφουμε τα φύλλα που περιέχουν τις βεβαιώσεις
   Delete_Sheets

End Sub
```

```
Private Sub Workbook_Open()

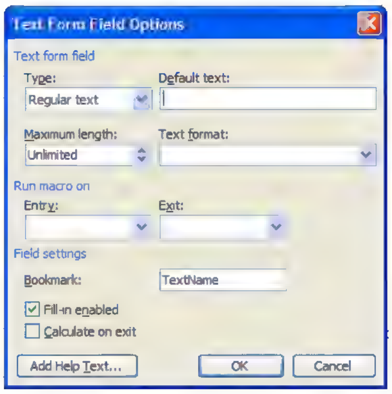
   ' Εμφανίζουμε την φόρμα σε κατάσταση modeless
   SplashScreen.Show 0

End Sub
```

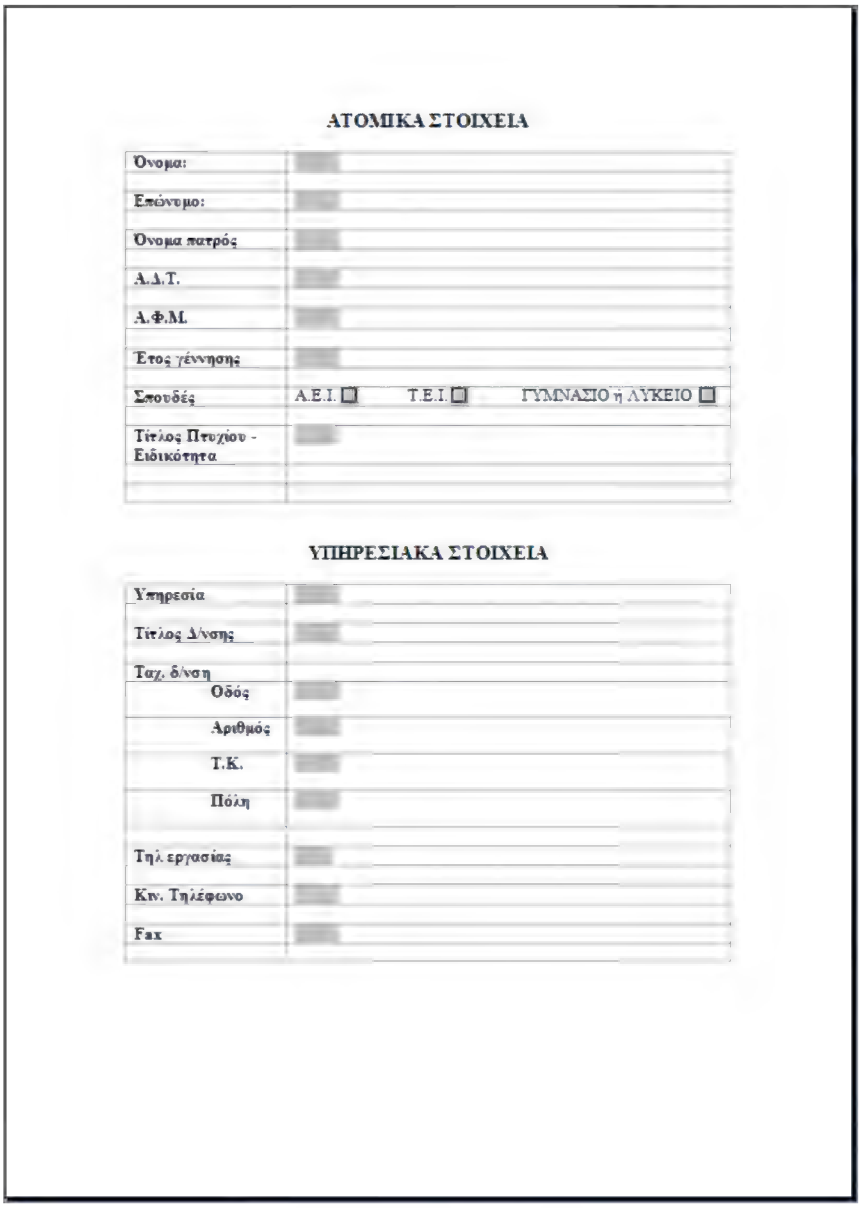
Ολοκληρωμένο παράδειγμα 2

Στο επόμενο παράδειγμα ανοίγουμε έγγραφα του Word, ανακτούμε από αυτά πληροφορίες και τις αποθηκεύουμε στο βιβλίο εργασίας. Χρησιμοποιούμε τη μέθοδο **GetOpenFileName** προκειμένου να επιλέξουμε τα έγγραφα του Word που θέλουλε να ανοίξουμε. Και σε αυτό το παράδειγμα πρέπει να έχουμε δηλώσει τη σχετική βιβλιοθήκη **Microsoft Word 11.0 Object Library**.

Στα έγγραφα του Word χρησιμοποιούμε **πεδία φόρμας** (**form fields**) για να αποθηκεύσουμε τις πληροφορίες που θέλουμε στη συνέχεια να ανακτήσουμε στο Excel. Τα πεδία που χρησιμοποιούμε είναι **TextForm** και **CheckBox**. Η ιδιότητα **Result** των σχετικών αντικειμένων επιστρέφει την τιμή που αποθηκεύσαμε στα **TextBox**, και 1 ή 0 στα **CheckBox**, ανάλογα με το να τα έχουμε επιλέξει ή όχι.



Εικόνα 83 Πεδίο φόρμας TextField στο έγγραφο του Word



ΑΤΟΜΙΚΑ ΣΤΟΙΧΕΙΑ

Όνομα:	
Επώνυμο:	
Όνομα πατρός	
Α.Δ.Τ.	
Α.Φ.Μ.	
Έτος γέννησης	
Σπουδές	Α.Ε.Ι. ☐ Τ.Ε.Ι. ☐ ΓΥΜΝΑΣΙΟ ή ΛΥΚΕΙΟ ☐
Τίτλος Πτυχίου - Ειδικότητα	

ΥΠΗΡΕΣΙΑΚΑ ΣΤΟΙΧΕΙΑ

Υπηρεσία	
Τίτλος Δ/νσης	
Ταχ. δ/νση	
Οδός	
Αριθμός	
Τ.Κ.	
Πόλη	
Τηλ εργασίας	
Κιν. Τηλέφωνο	
Fax	

Εικόνα 84 Το έγγραφο του Word

Ο παρακάτω κώδικας περιέχεται στο module:

```
Option Explicit
 Dim cnlprcs As Boolean
```

```
Sub Open_folder()

    Dim filt As String
    Dim filterindex As Integer
    Dim filename As Variant
    Dim title As String
    Dim i As Integer
    Dim Msg As String
    Dim wrkbk As Workbook
```

```
    filt = "Word files (*.doc),*.doc"
    filterindex = 1

    title = "Επιλέξτε τα αρχεία word που επιθυμείτε να επεξεργαστείτε"

    filename = Application.GetOpenFilename _
       (filefilter:=filt, _
        filterindex:=filterindex, _
        title:=title, _
        MultiSelect:=True)

    If Not IsArray(filename) Then
       MsgBox "Δεν επιλέχτηκε κανένα αρχείο"
       Exit Sub
    Else
       For i = LBound(filename) To UBound(filename)
          Msg = Msg & filename(i) & vbCrLf
       Next i
       MsgBox "Επιλέξατε τα παρακάτω αρχεία: " & vbCrLf & Msg
    End If

    cnlprcs = False

    For i = LBound(filename) To UBound(filename)
       If Not cnlprcs Then
          Process_Word_Document_2 filename(i)
       Else
          MsgBox "Ακυρώσατε την επεξεργασία"
       End If
    Next i

    Application.StatusBar = False
    ThisWorkbook.Worksheets(1).Columns("A:T").AutoFit


End Sub
```

```
Sub Process_Word_Document_2(filename As Variant)

    Dim wrdapp As New Word.Application
    Dim wrksht As Worksheet
    Dim wrddoc As Word.Document
    Dim nam As String
    Dim surname As String
    Dim fathername As String
    Dim adt As String
    Dim afm As String
    Dim birthdate As String
    Dim aei As Variant
    Dim tei As Variant
    Dim lyceum As Variant
    Dim educationtitle As String
```

```
Dim service As String
Dim directory As String
Dim street As String
Dim streetno As String
Dim tk As String
Dim city As String
Dim worktel As String
Dim mobiletel As String
Dim fax As String

Dim path_filename As String
Dim fname As String
Dim fnm_dp As Integer

Dim i As Integer

' δημιουργούμε ένα αντικείμενο της εφαρμογής word
On Error Resume Next
Set wrdapp = GetObject(, "Word.Application")

'
If wrdapp Is Nothing Then
   Set wrdapp = GetObject("", "Word.Application")
End If

On Error GoTo Error_Handler_1

' προσθέτουμε ένα νέο έγγραφο μη ορατό, απενεργοποιούμε τα μηνύματα που τυχόν
' εμφανίζει το Word και ανοίγουμε το έγγραφο μόνο για ανάγνωση
   With wrdapp
      .Visible = False
      .AutomationSecurity = msoAutomationSecurityLow
      .DisplayAlerts = wdAlertsNone
      .Documents.Open filename, ReadOnly:=True
   End With
   Set wrddoc = wrdapp.ActiveDocument
   ' Debug.Print wrddoc.name
   Excel.Application.StatusBar = "Επεξεργασία του αρχείου: " & _
   wrdapp.ActiveDocument.name

On Error GoTo Error_Handler_2

' Ελέγχουμε αν το έγγραφο που ανοίξαμε έχει τουλάχιστον ένα πεδίο
If wrdapp.ActiveDocument.FormFields.Count <> "0" Then
   ' Για κάθε πεδίο φόρμας αποθηκεύομε την τιμή του σε μία μεταβλητή
   nam = wrdapp.ActiveDocument.FormFields("TextName").Result
   surname = wrdapp.ActiveDocument.FormFields("TextSurName").Result
   fathername = wrdapp.ActiveDocument.FormFields("TextFatherName").Result
   adt = wrdapp.ActiveDocument.FormFields("TextADT").Result
   afm = wrdapp.ActiveDocument.FormFields("TextAFM").Result
   birthdate = wrdapp.ActiveDocument.FormFields("TextBirthDate").Result
   aei = wrdapp.ActiveDocument.FormFields("CheckAEI").Result
   tei = wrdapp.ActiveDocument.FormFields("CheckTEI").Result
   lyceum = wrdapp.ActiveDocument.FormFields("CheckLyceum").Result
```

```
        educationtitle = wrdapp.ActiveDocument.FormFields("TextEducation").Result
        service = wrdapp.ActiveDocument.FormFields("TextService").Result
        directory = wrdapp.ActiveDocument.FormFields("TextDirectory").Result
        street = wrdapp.ActiveDocument.FormFields("TextStreet").Result
        streetno = wrdapp.ActiveDocument.FormFields("TextStreetNo").Result
        city = wrdapp.ActiveDocument.FormFields("TextCity").Result
        worktel = wrdapp.ActiveDocument.FormFields("TextWorkTel").Result
        mobiletel = wrdapp.ActiveDocument.FormFields("TextMobileTel").Result
        fax = wrdapp.ActiveDocument.FormFields("TextFAX").Result

        ' Βρίσκουμε το πρώτο κελί στη στήλη A που δεν έχει περιεχόμενο
        i = 1
        Do Until IsEmpty(Cells(i, 1))
           i = i + 1
        Loop

        ' Εισάγουμε τις τιμές των μεταβλητών στα κελιά του φύλλο εργασίας
        Cells(i, 1) = nam
        Cells(i, 2) = surname
        Cells(i, 3) = fathername
        Cells(i, 4) = adt
        Cells(i, 4).NumberFormat = "0"
        Cells(i, 5) = afm
        Cells(i, 5).NumberFormat = "0"
        Cells(i, 6) = birthdate
        Cells(i, 7) = aei
        Cells(i, 8) = tei
        Cells(i, 9) = lyceum
        Cells(i, 10) = educationtitle
        Cells(i, 11) = service
        Cells(i, 12) = directory
        Cells(i, 13) = street
        Cells(i, 14) = streetno
        Cells(i, 14).NumberFormat = "0"
        Cells(i, 15) = city
        Cells(i, 16) = worktel
        Cells(i, 16).NumberFormat = "0"
        Cells(i, 17) = mobiletel
        Cells(i, 17).NumberFormat = "0"
        Cells(i, 18) = fax
        Cells(i, 18).NumberFormat = "0"
        Cells(i, 19) = Get_Only_Filename(filename)
    Else
    ' Αν το έγγραφο του Word δεν περιέχει κανένα πεδίο φόρμας
    ' εμφανίζουμε σχετικό μήνυμα
       MsgBox "Το αρχείο " & StrReverse(fname) & _
       "δεν είναι κατάλληλο αρχείο εισόδου" & vbCrLf & _
       "Δεν περιέχει κανένα πεδίο φόρμας"
    End If

    With wrdapp
        .ActiveDocument.Close
    End With
    wrdapp.Quit
    Set wrdapp = Nothing
```

```
    Application.StatusBar = False

    Exit Sub

Error_Handler_1:
    For Each wrddoc In wrdapp.Documents
        wrddoc.Close
    Next wrddoc
    wrdapp.Quit
    Set wrdapp = Nothing
    Application.StatusBar = False
    Exit Sub

Error_Handler_2:
    MsgBox "Κάποιο πεδίο δεν υπάρχει"
    For Each wrddoc In wrdapp.Documents
        wrddoc.Close
    Next wrddoc
    wrdapp.Quit
    Set wrdapp = Nothing
    Application.StatusBar = False
    Exit Sub

End Sub
```

```
Sub SaveAs_xls()

    Dim filt As String
    Dim filterindex As Integer
    Dim filename As Variant
    Dim title As String
    Dim i As Integer
    Dim wrkbk As Workbook
    Dim swrkbk As Workbook
    Dim eaddr As String

    filt = "Excel files (*.xls),*.xls"
    filterindex = 1

    filename = Application.GetSaveAsFilename(filefilter:=filt, title:=title, _
    filterindex:=filterindex)

    ThisWorkbook.Worksheets(1).Cells(2, 1).Activate
    Range(ActiveCell, ActiveCell.Offset(0, 18).End(xlDown)).Copy
    Workbooks.Open filename

    Debug.Print filename
    For Each wrkbk In Workbooks
        Debug.Print wrkbk.name
        If wrkbk.name = Get_Only_Filename(filename) Then
            Set swrkbk = wrkbk
        End If
    Next wrkbk
```

```
    i = 1
    Do Until IsEmpty(swrkbk.Worksheets(1).Cells(i, 1))
       i = i + 1
    Loop
    eaddr = Cells(i, 1).Address
    Debug.Print eaddr
    swrkbk.Worksheets(1).Range(eaddr).PasteSpecial
    swrkbk.Save
    swrkbk.Close

End Sub
```

```
Function Get_Only_Filename(filename As Variant) As String

    Dim fnm_dp As Integer
    Dim fname As String
    Dim path_filename As String

    path_filename = StrReverse(filename)
    fnm_dp = InStr(1, path_filename, "\")
    fname = Left(path_filename, fnm_dp - 1)
    Get_Only_Filename = StrReverse(fname)

End Function
```

```
Sub Cancel_and_exit()

    ThisWorkbook.Worksheets(1).Range("A2", ActiveCell.Offset(0, 18). _
    End(xlDown)).ClearContents

End Sub
```

Ο κώδικας της φόρμας, που φαίνεται στην επόμενη εικόνα, είναι ο παρακάτω:

```
Private Sub CBCancel_Click()

    Cancel_and_exit

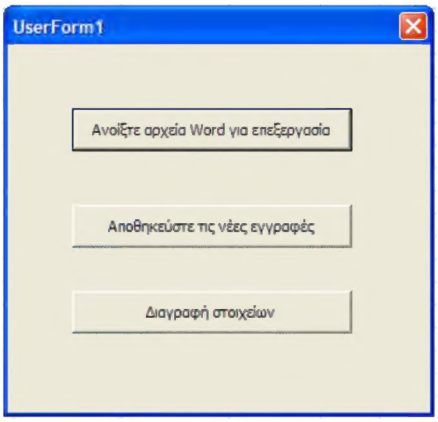
End Sub
```

```
Private Sub CBOpen_Click()

    Open_folder
```

```
End Sub
```

```
Private Sub CBSaveAs_Click()
    SaveAs_xls
End Sub
```



Εικόνα 85 Η φόρμα του παραδείγματος

Στο βιβλίο εργασίας υπάρχουν μία διαδικασία, που εμφανίζει τη φόρμα.

```
Private Sub Workbook_Open()
    UserForm1.Show 0
End Sub
```

Στο πρώτο φύλλο εργασίας υπάρχει ένα **CommandButton** που καλεί τη φόρμα:

```
Private Sub CommandButton1_Click()
    UserForm1.Show 0
End Sub
```

Αποστολή email μέσω Outlook

Στο επόμενο παράδειγμα αποστέλλουμε email μέσω της εφαρμογής Outlook. Και εδώ πρέπει να έχουμε δηλώσει τη σχετική βιβλιοθήκη **Microsoft Outlook 11.0 Object Library,** στο μενού **Tools – References**. Ο κώδικας αναζητεί στη στήλη B διευθύνσεις ηλεκτρονικού ταχυδρομίου, στη μορφή

"*@*" και συντάσει ένα μήνυμα με σταθερό κείμενο και τα στοιχεία που βρίσκονται στα δύο κελιά της ανίστοιχης γραμμής που περιέχει την διεύθυνση.

	A	B	C	D
1	**Name**	**Email**	**Bonus**	
2	Θεόδωρος Παναγόπουλος	theodor_panagopoulos@gmail.gr	1.000,00 €	
3	Γεώργιος Τεστίδης	test@yahoo.com	2.000,00 €	
4				

Εικόνα 86 Τα στοιχεία που είναι αποθηκευμένα στο φύλλο εργασίας και χρησιμοποιεί ο κώδικας του παραδείγματος για την αποστολή email

Ο παρακάτω κώδικας περιέχεται στο module:

```
Sub SendEmail_with_Outlook()

Dim otlkapp As Outlook.Application
Dim mitem As Outlook.MailItem
Dim cell As Range
Dim subj As String
Dim emailaddr As String
Dim recipient As String
Dim bonus As String
Dim msg As String

' Δημιουργούμε ένα νέο αντικείμενο της εφαρμογής ηλεκτρονικής αλληλογραφίας
Set otlkapp = New Outlook.Application

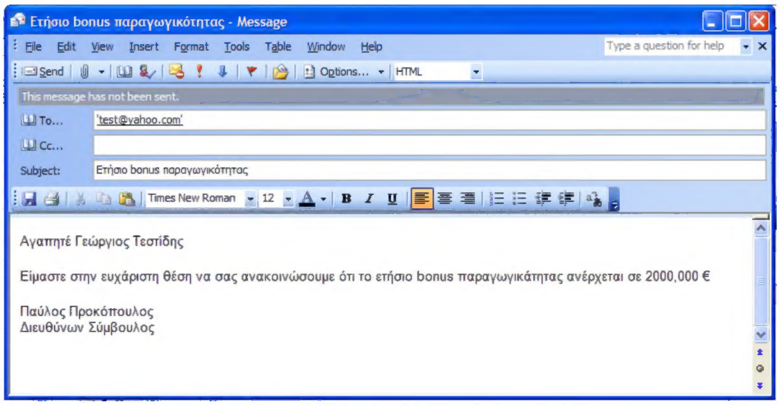
' Αναζητούμε διευθύνσεις ηλεκτρονικής αλληλογραφίας
' Η έκφραση Columns("B").Cells.SpecialCells(xlCellTypeConstants)
' αναζητά κελιά που περιέχουν σταθερές
For Each cell In Columns("B").Cells.SpecialCells(xlCellTypeConstants)
   If cell.Value Like "*@*" Then
      ' Αντιγράφουμε τα στοιχεία
      subj = "Ετήσιο bonus παραγωγικότητας"
      recipient = cell.Offset(0, -1).Value
      emailaddr = cell.Value
      bonus = Format(cell.Offset(0, 1).Value, "0.000, E")

      ' Συνθέτουμε το σώμα του μηνύματος
      msg = "Αγαπητέ " & Recipient & vbCrLf & vbCrLf
      msg = msg & "Είμαστε στην ευχάριστη θέση να σας ανακοινώσουμε " & _
      "ότι το ετήσιο bonus παραγωγικάτητας ανέρχεται σε "
      msg = msg & Bonus & "€" & vbCrLf & vbCrLf
      msg = msg & "Παύλος Προκόπουλος" & vbCrLf
      msg = msg & "Διευθύνων Σύμβουλος"

      ' Δημιουργούμε το μήνυμα και το αποστέλλουμε
      Set mitem = OutlookApp.CreateItem(olMailItem)
      With mitem
```

```
            .To = emailaddr
            .Subject = subj
            .Body = msg
            .Send
        End With
    End If
Next

End Sub
```



Εικόνα 87 Το μήνυμα που αποστέλλει το Outlook

Βιβλιογραφία

1. Excel 2003 Power Programming, John Walkenbach, Wwilley Publishing, Inc.
2. Excel Programming, Your visual blueprint for creating interactive spreadsheets, Jinjer Simon, Hungry Minds, Inc.
3. Excel 2003, Prgrammers Reference, Paul Kimmel, Stephen Bullen, John Green, Rob Bovey, Robert Rosenberg, Wiley Publishibg, Inc.
4. Definitive Guide to Excel VBA, Michael Kofler, Apress